TCP/IP Network, Transport and Application Layers

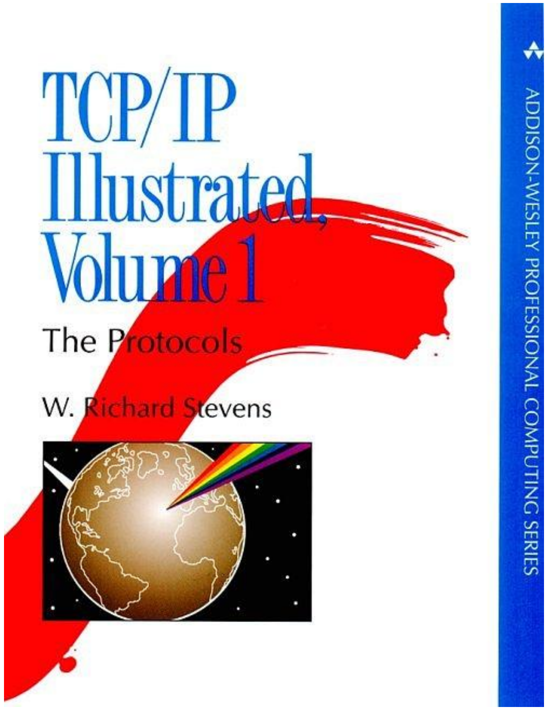
CIS 81 and CST 311

Rick Graziani

Spring 2006

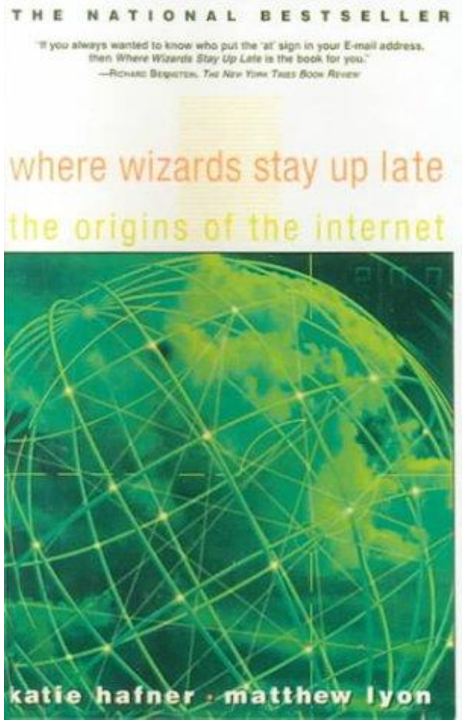
- It is important for networking professionals to have a very good understanding of TCP/IP.
- Various devices communicate using the multiple protocols of the TCP/IP protocol suite.
- A networking professional needs to know how these protocols function and interact with each other in order to properly understand, analyze and troubleshoot networking issues.
- This chapter is only an introduction to this information.
- I strongly suggest taking a separate course in the TCP/IP protocol suite, in addition to system administration courses such as those for Microsoft Windows (MCSE/MCSA) or Unix/Linux.
- The majority of this presentation is taken directly from the on-line curriculum (present and past) – however there are a few mistakes or misconceptions in the on-line curriculum which is addressed in this presentation.
- Many of the concepts in the presentation are missing some important details to keep the amount of information to a reasonable limit – Again I suggest taking a course on TCP/IP protocol suite.
- Also, two other presentations are included on my web site:
  - ARP
  - ICMP – Understanding ping and trace

Important and Interesting Reading


TCP/IP Illustrated, Volume 1
The Protocols
W. Richard Stevens
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

TCP/IP Illustrated, Vol. 1
W. Richard Stevens
Addison-Wesley Pub Co
ISBN: 0201633469

- Although, published in 1994, written by the late Richard Stevens, it is still regarded as the definitive book on TCP/IP.


THE NATIONAL BESTSELLER
"If you always wanted to know who put the 'at' sign in your E-mail address, then Where Wizards Stay Up Late is the book for you."
—Richard Bernstein, The New York Times Book Review
where wizards stay up late
the origins of the internet
katie hafner · matthew lyon

Where Wizards Stay Up Late
Katie Hafner and Matthew Lyon
ISBN 0613181530

- Very enjoyable reading and you do not have to be a networking geek to enjoy it!
- National Bestseller

Topics

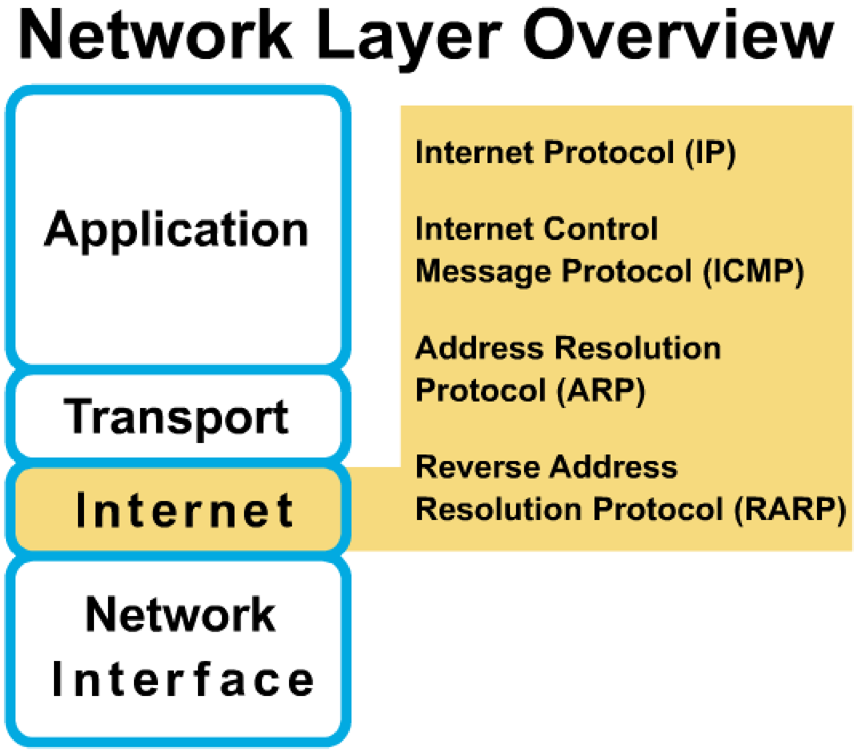
Layer 3, Network Layer Concepts

- TCP/IP and the Internet Layer
- Diagram the IP datagram
- Internet Control Message Protocol (ICMP)

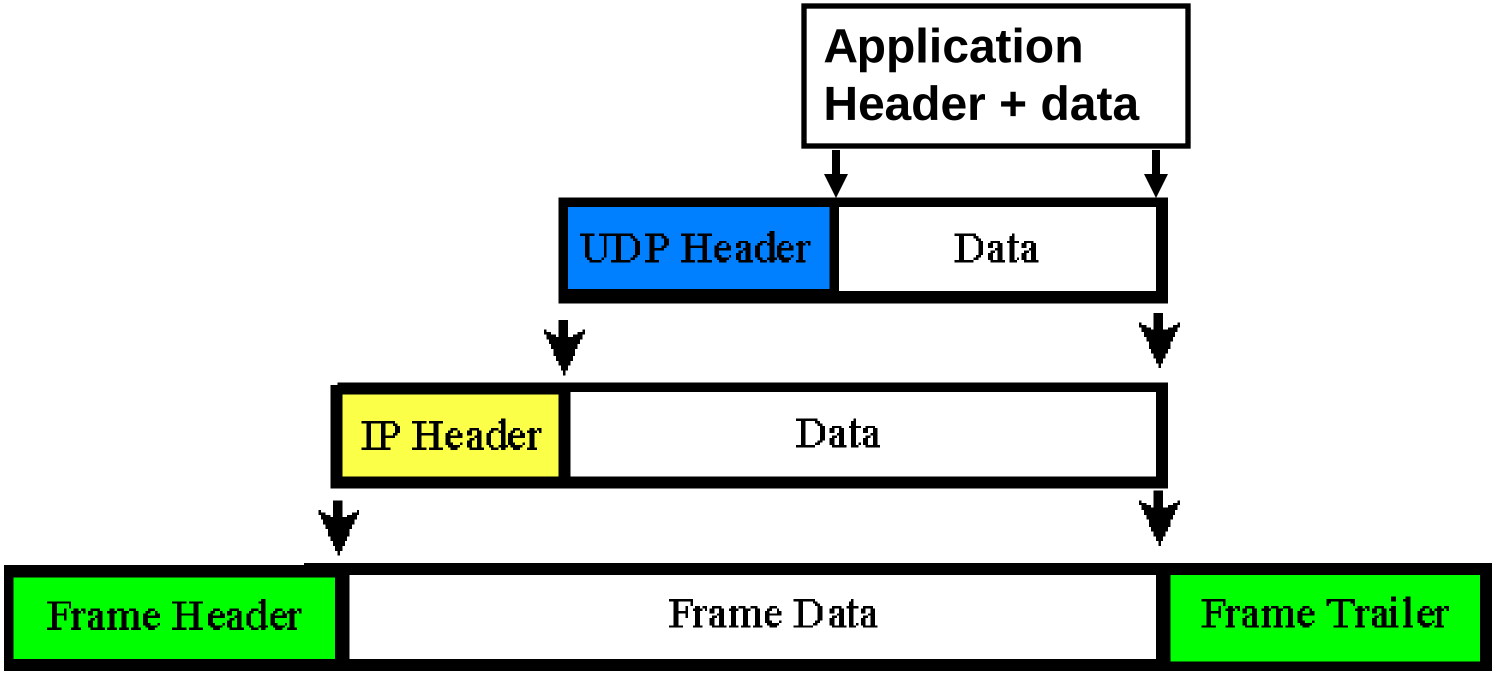
TCP/IP protocol stack and the transport layer

- TCP and UDP segment format
- TCP and UDP port numbers
- TCP three-way handshake/open connection
- TCP simple acknowledgment and windowing

Layer 3: TCP/IP Network Layer


Network Layer Overview
Application
Transport
Internet
Network Interface
Internet Protocol (IP)
Internet Control Message Protocol (ICMP)
Address Resolution Protocol (ARP)
Reverse Address Resolution Protocol (RARP)

- The Internet layer of the TCP/IP stack corresponds to the network layer of the OSI model.
- Each layer is responsible for getting packets through a network using software addressing.

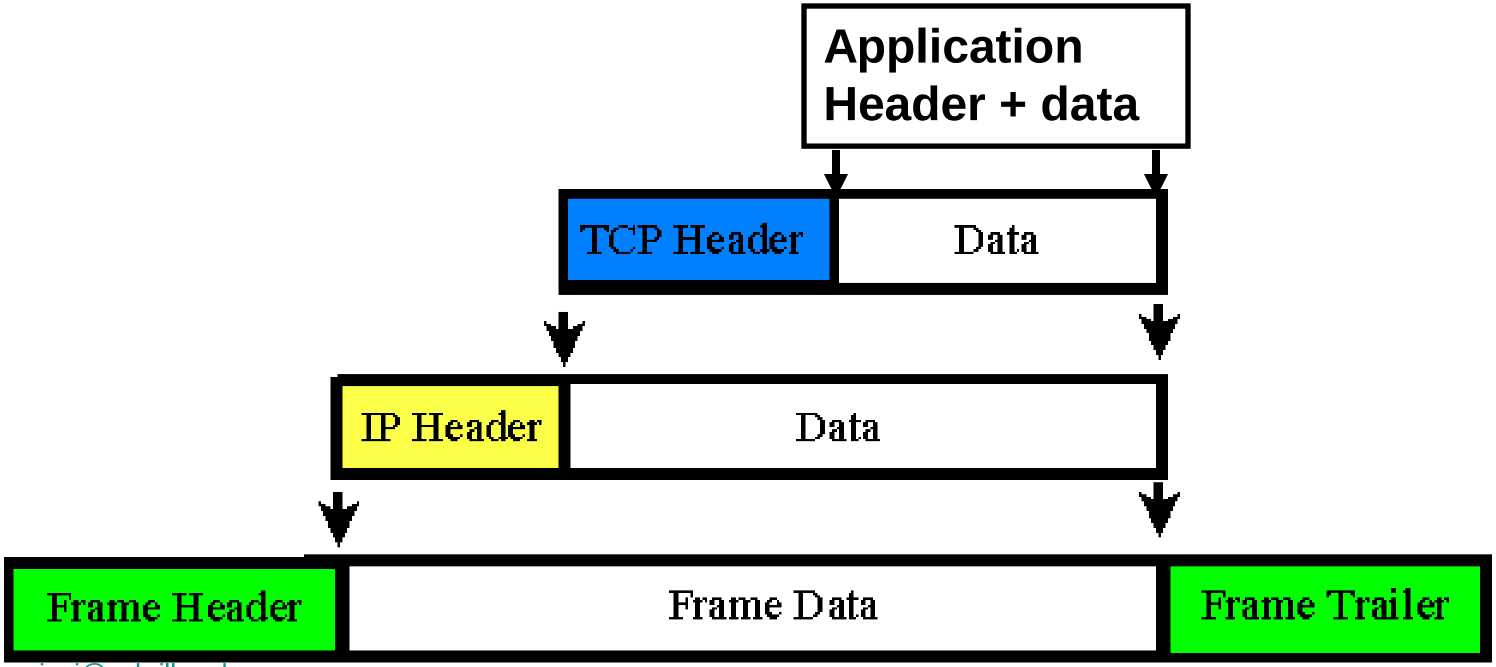


Application Header + data

UDP Header | Data

IP Header | Data

Frame Header | Frame Data | Frame Trailer



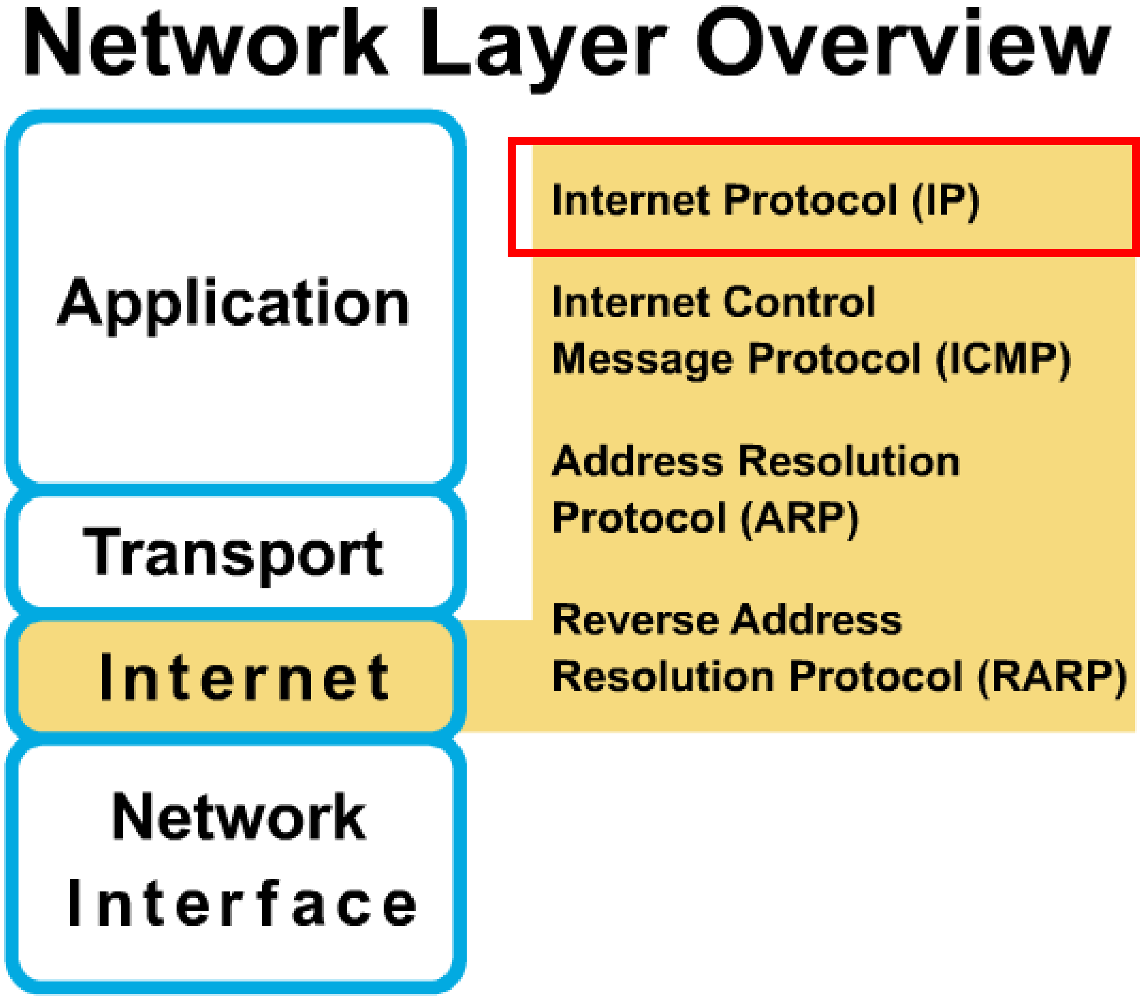
Application Header + data

TCP Header | Data

IP Header | Data

Frame Header | Frame Data | Frame Trailer

IP – Internet Protocol



Network Layer Overview

Application

Transport

Internet

Network Interface

Internet Protocol (IP)

Internet Control Message Protocol (ICMP)

Address Resolution Protocol (ARP)

Reverse Address Resolution Protocol (RARP)

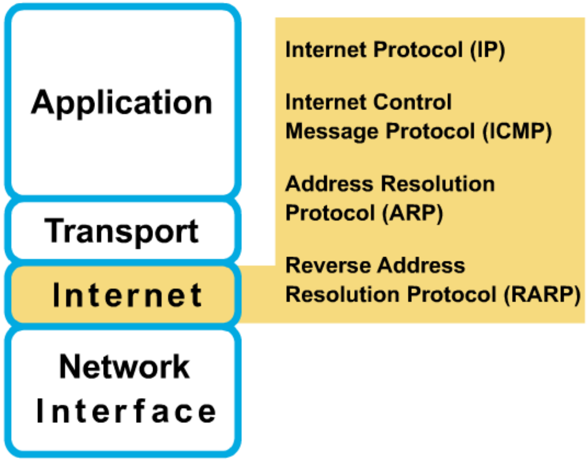
IP Packet (Data Gram) Header

<table>
<tr><th>0</th><th></th><th>15</th><th>16</th><th>31</th></tr>
<tr><td>4-bit Version</td><td>4-bit Header Length</td><td>8-bit Type Of Service (TOS)</td><td colspan="2">16-bit Total Length (in bytes)</td></tr>
<tr><td colspan="3">16-bit Identification</td><td>3-bit Flags</td><td>13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8 bit Time To Live TTL</td><td>8-bit Protocol</td><td colspan="2">16-bit Header Checksum</td></tr>
<tr><td colspan="5">32-bit Source IP Address</td></tr>
<tr><td colspan="5">32-bit Destination IP Address</td></tr>
</table>

Options (if any)

Data

Network Layer Overview



Application

Transport

Internet

Network Interface

Internet Protocol (IP)

Internet Control Message Protocol (ICMP)

Address Resolution Protocol (ARP)

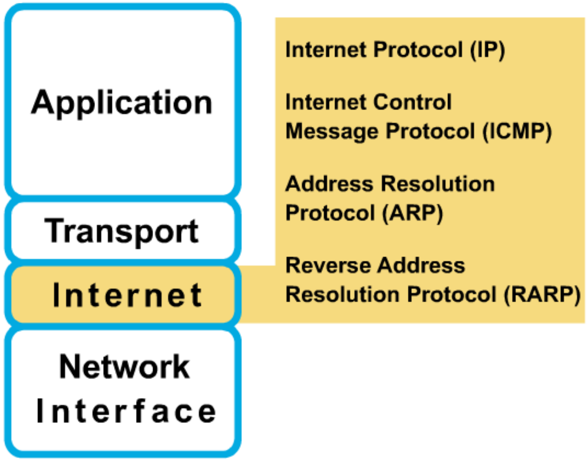
Reverse Address Resolution Protocol (RARP)

IP Packet (Data Gram) Header

<table>
<tr><th>0</th><th></th><th>15</th><th>16</th><th>31</th></tr>
<tr><td>4-bit Version</td><td>4-bit Header Length</td><td>8-bit Type Of Service (TOS)</td><td colspan="2">16-bit Total Length (in bytes)</td></tr>
<tr><td colspan="3">16-bit Identification</td><td>3-bit Flags</td><td>13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8 bit Time To Live TTL</td><td>8-bit Protocol</td><td colspan="2">16-bit Header Checksum</td></tr>
<tr><td colspan="5">32-bit Source IP Address</td></tr>
<tr><td colspan="5">32-bit Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any)</td></tr>
<tr><td colspan="5">Data</td></tr>
</table>

- ***VERS*** -- version number
- ***HLEN*** -- header length, in 32-bit words
- ***type of service*** -- how the datagram should be handled
- ***total length*** -- total length (header + data)
- ***identification, flags, flag offset*** -- provides fragmentation of datagrams to allow differing MTUs in the internetwork

Network Layer Overview


Application
Transport
Internet
Network Interface
Internet Protocol (IP)
Internet Control Message Protocol (ICMP)
Address Resolution Protocol (ARP)
Reverse Address Resolution Protocol (RARP)

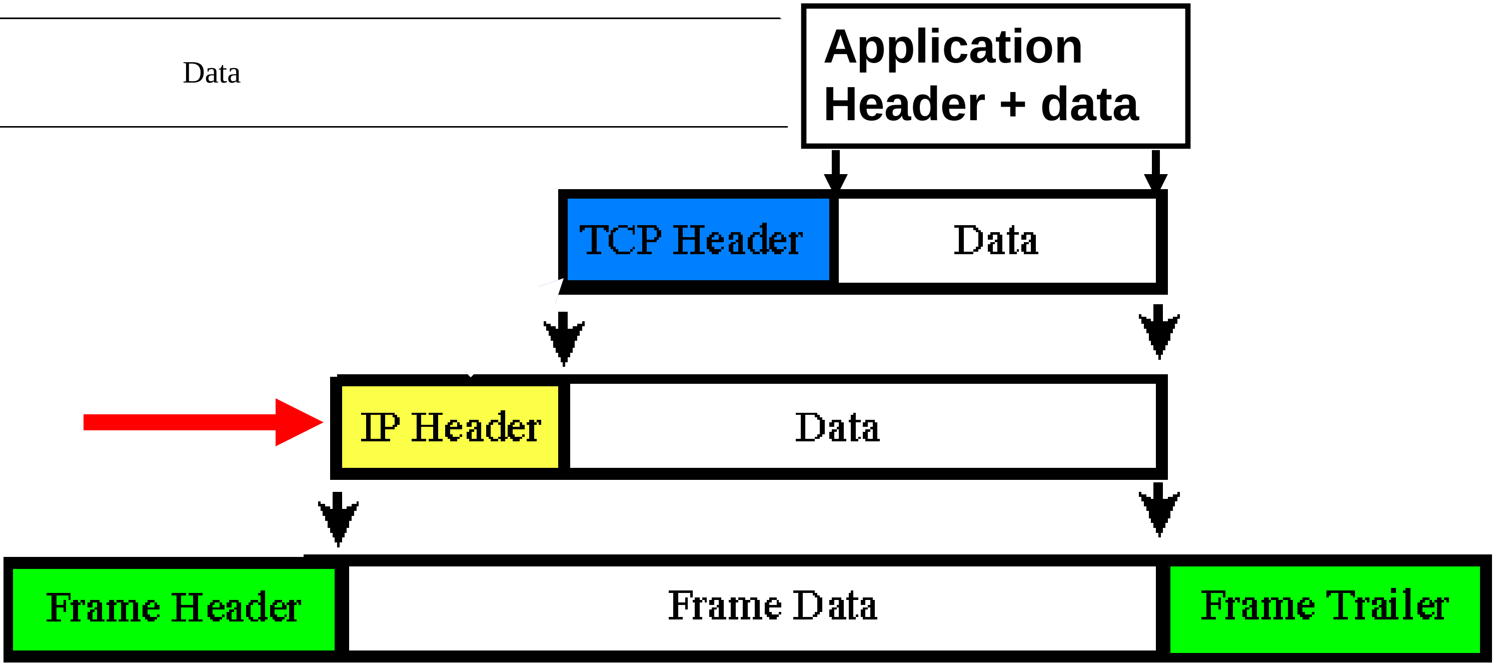
IP Packet (Data Gram) Header

0		15	16	31
4-bit Version	4-bit Header Length	8-bit Type Of Service (TOS)	16-bit Total Length (in bytes)	
16-bit Identification			3-bit Flags	13-bit Fragment Offset
8 bit Time To Live TTL		8-bit Protocol	16-bit Header Checksum	
32-bit Source IP Address				
32-bit Destination IP Address				
Options (if any)				
Data				

- ***TTL*** -- Time-To-Live
- ***protocol*** -- the upper-layer (Layer 4) protocol sending the datagram
- ***header checksum*** -- an integrity check on the header
- ***source IP address*** *and* ***destination IP address*** -- 32-bit IP addresses
- ***IP options*** -- network testing, debugging, security, and other options
- **Data** – Upper layer headers and data

IP Header

0		15	16	31
4-bit Version	4-bit Header Length	8-bit Type Of Service (TOS)	16-bit Total Length (in bytes)	
16-bit Identification			3-bit Flags	13-bit Fragment Offset
8 bit Time To Live TTL		8-bit Protocol	16-bit Header Checksum	
32-bit Source IP Address				
32-bit Destination IP Address				
Options (if any)				
Data				



Application Header + data

TCP Header

Data

IP Header

Data

Frame Header

Frame Data

Frame Trailer

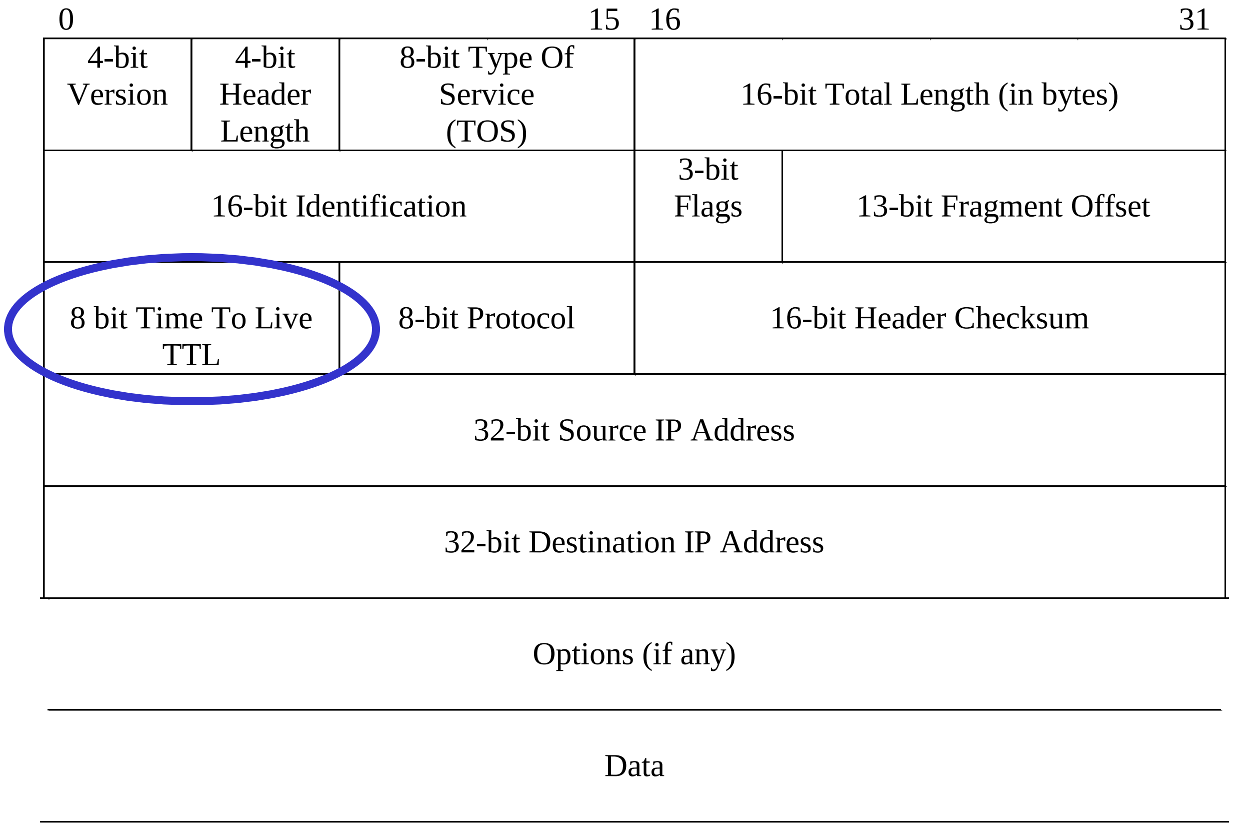
IP’s TTL – Time To Live field

IP Header

<table>
<tr><td colspan="3">0</td><td>15</td><td colspan="3">16</td><td>31</td></tr>
<tr><td colspan="2">4-bit Version</td><td colspan="2">4-bit Header Length</td><td colspan="4">8-bit Type Of Service (TOS)</td><td colspan="8">16-bit Total Length (in bytes)</td></tr>
<tr><td colspan="8">16-bit Identification</td><td colspan="2">3-bit Flags</td><td colspan="6">13-bit Fragment Offset</td></tr>
<tr><td colspan="4">8 bit Time To Live TTL</td><td colspan="4">8-bit Protocol</td><td colspan="8">16-bit Header Checksum</td></tr>
<tr><td colspan="16">32-bit Source IP Address</td></tr>
<tr><td colspan="16">32-bit Destination IP Address</td></tr>
<tr><td colspan="16">Options (if any)</td></tr>
<tr><td colspan="16">Data</td></tr>
</table>

IP's TTL – Time To Live field

IP Header

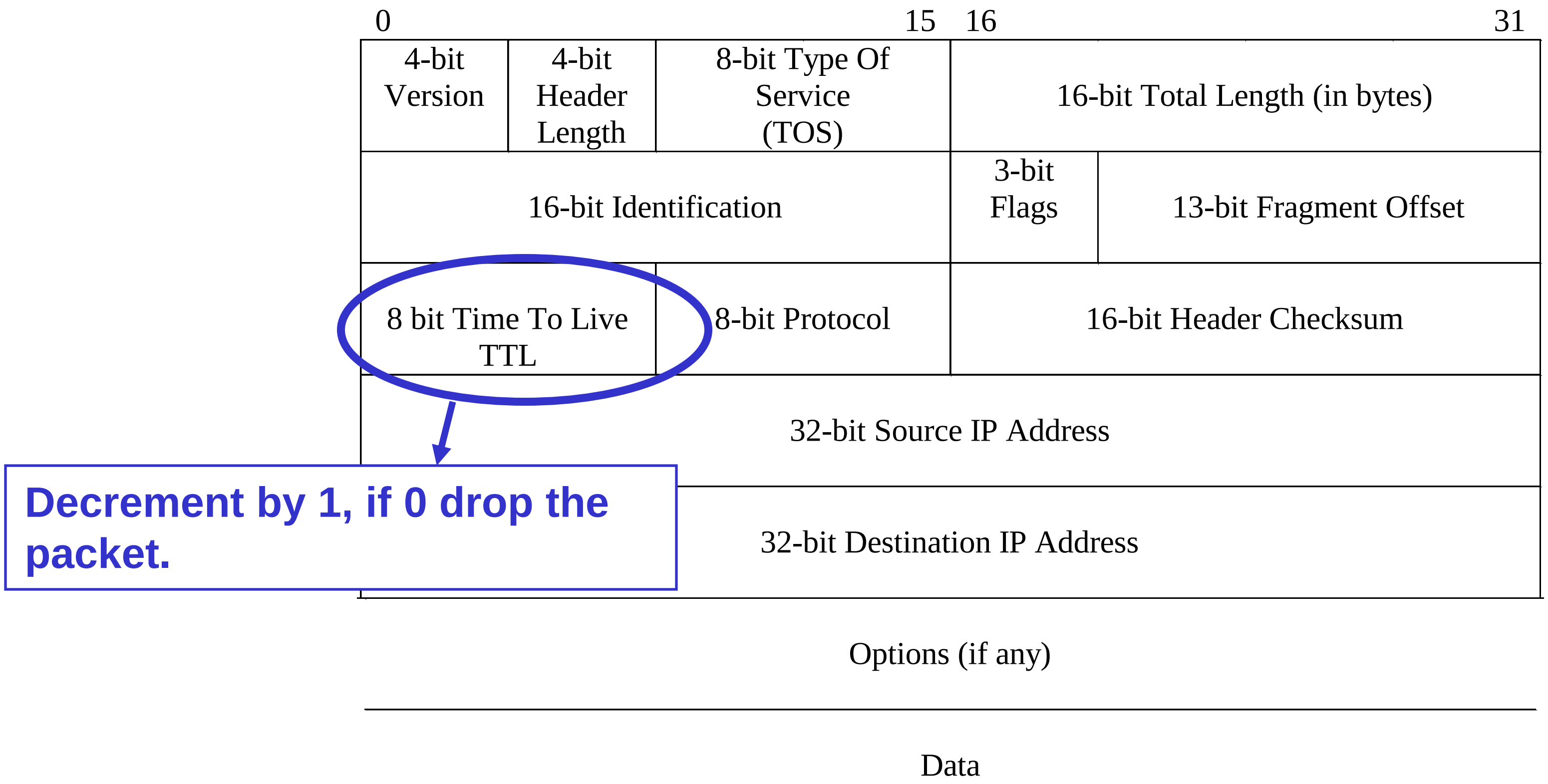


<table>
<tr><td>0</td><td></td><td>15</td><td>16</td><td>31</td></tr>
<tr><td>4-bit Version</td><td>4-bit Header Length</td><td>8-bit Type Of Service (TOS)</td><td colspan="2">16-bit Total Length (in bytes)</td></tr>
<tr><td colspan="3">16-bit Identification</td><td>3-bit Flags</td><td>13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8 bit Time To Live TTL</td><td>8-bit Protocol</td><td colspan="2">16-bit Header Checksum</td></tr>
<tr><td colspan="5">32-bit Source IP Address</td></tr>
<tr><td colspan="5">32-bit Destination IP Address</td></tr>
<tr><td colspan="5">Options (if any)</td></tr>
<tr><td colspan="5">Data</td></tr>
</table>

- When a packet is first generated a value is entered into the TTL field.
- Originally, the TTL field was the number of seconds, but this was difficult to implement and rarely supported.
- Now, the TTL is now set to a specific value which is then decremented by each router.

IP's TTL – Time To Live field

IP Header



0 15 16 31

4-bit Version | 4-bit Header Length | 8-bit Type Of Service (TOS) | 16-bit Total Length (in bytes)

16-bit Identification | 3-bit Flags | 13-bit Fragment Offset

8 bit Time To Live TTL | 8-bit Protocol | 16-bit Header Checksum

32-bit Source IP Address

32-bit Destination IP Address

Options (if any)

Data

Decrement by 1, if 0 drop the packet.

- If the router decrements the TTL field to 0, it will then drop the packet (unless the packet is destined specifically for the router, I.e. ping, telnet, etc.).
- Common operating system TTL values are:
  - UNIX: **255**
  - Linux: **64 or 255** depending upon vendor and version
  - Microsoft Windows 95: **32**
  - Other Microsoft Windows operating systems: **128**

http://www.switch.ch/docs/ttl_default.html

TTL Overview - Disclaimer:

The following list is a best effort overview of some widely used TCP/IP stacks. The information was provided by vendors and many helpful system administrators. We would like to thank all these contributors for their precious help ! SWITCH cannot, however, take any responsibility that the provided information is correct. Furthermore, SWITCH cannot be made liable for any damage that may arise by the use of this information.

OS Version	"safe"	tcp_ttl	udp_ttl
AIX	n	60	30
DEC Pathworks V5	n	30	30
FreeBSD 2.1R	y	64	64
HP/UX 9.0x	n	30	30
HP/UX 10.01	y	64	64
Irix 5.3	y	60	60
Irix 6.x	y	60	60
Linux	y	64	64
MacOS/MacTCP 2.0.x	y	60	60
OS/2 TCP/IP 3.0	y	64	64
OSF/1 V3.2A	n	60	30
Solaris 2.x	y	255	255
SunOS 4.1.3/4.1.4	y	60	60
Ultrix V4.1/V4.2A	n	60	30
VMS/Multinet	y	64	64
VMS/TCPware	y	60	64
VMS/Wollongong 1.1.1.1	n	128	30
VMS/UCX (latest rel.)	y	128	128
MS WfW	n	32	32
MS Windows 95	n	32	32
MS Windows NT 3.51	n	32	32
MS Windows NT 4.0	y	128	128

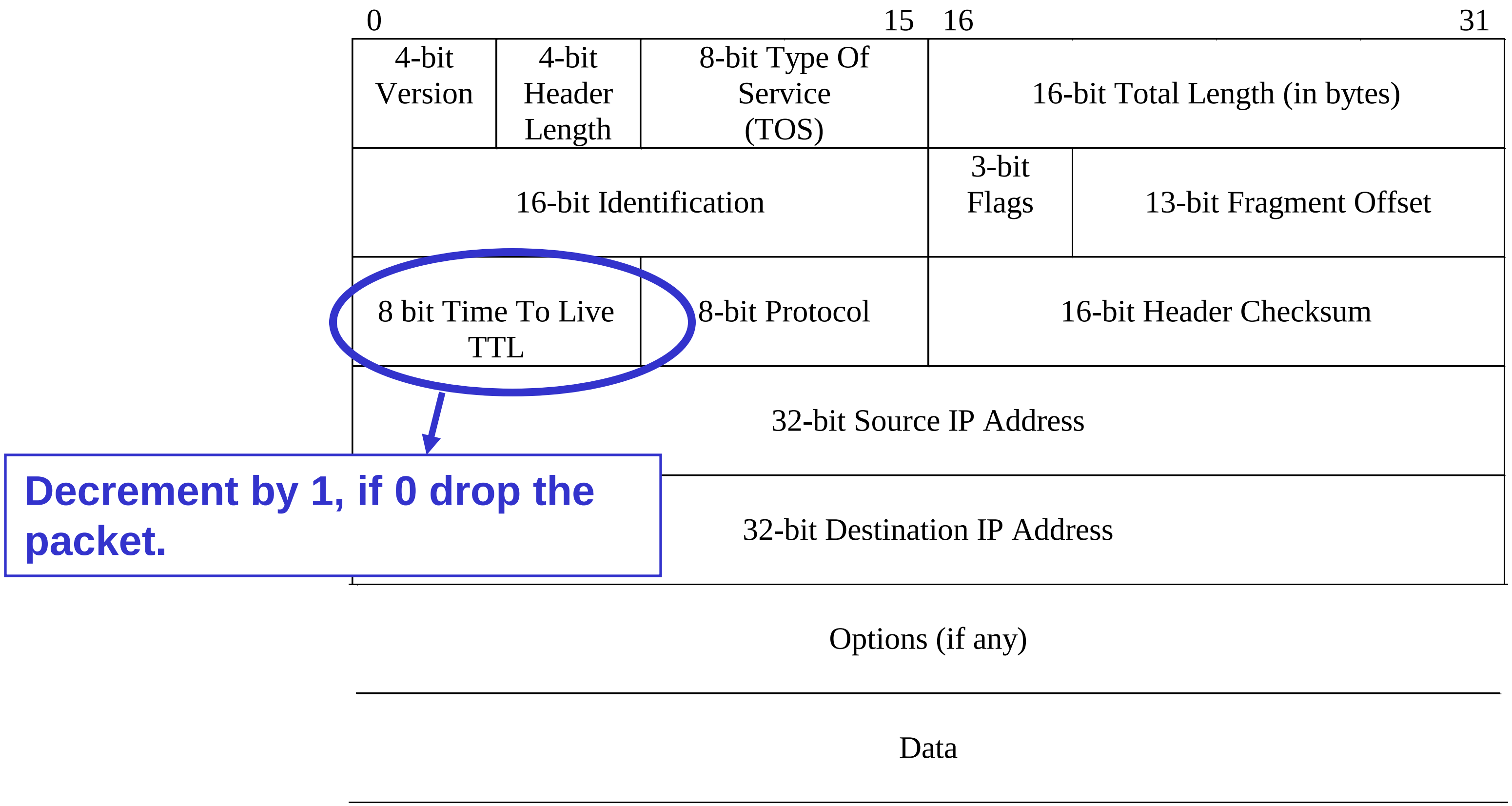
Assigned Numbers (RFC 1700, J. Reynolds, J. Postel, October 1994):

> IP TIME TO LIVE PARAMETER
>
> The current recommended default time to live (TTL) for the Internet Protocol (IP) is 64.

Safe: TCP and UDP initial TTL values should be set to a "safe" value of at least 60 today.

IP’s TTL – Time To Live field

IP Header

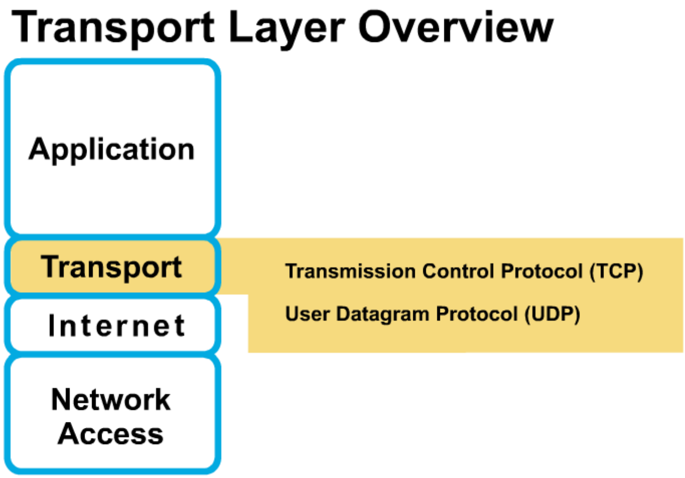


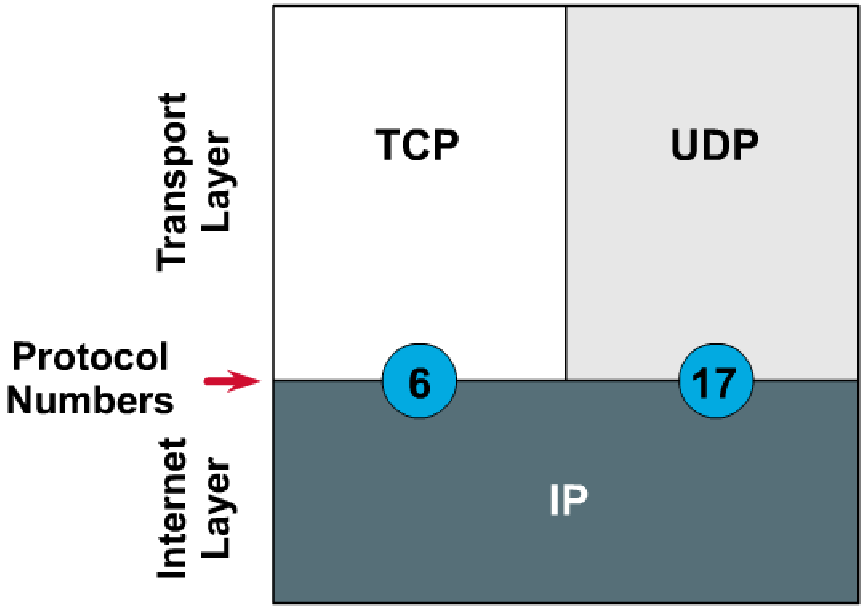
<table>
<tr><td colspan="4">0   15</td><td colspan="4">16   31</td></tr>
<tr><td>4-bit Version</td><td>4-bit Header Length</td><td colspan="2">8-bit Type Of Service (TOS)</td><td colspan="4">16-bit Total Length (in bytes)</td></tr>
<tr><td colspan="4">16-bit Identification</td><td>3-bit Flags</td><td colspan="3">13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8 bit Time To Live TTL</td><td colspan="2">8-bit Protocol</td><td colspan="4">16-bit Header Checksum</td></tr>
<tr><td colspan="8">32-bit Source IP Address</td></tr>
<tr><td colspan="8">32-bit Destination IP Address</td></tr>
<tr><td colspan="8">Options (if any)</td></tr>
<tr><td colspan="8">Data</td></tr>
</table>

Decrement by 1, if 0 drop the packet.

- The idea behind the TTL field is that IP packets can not travel around the Internet forever, from router to router.
- Eventually, the packet’s TTL which reach 0 and be dropped by the router, even if there is a routing loop somewhere in the network.

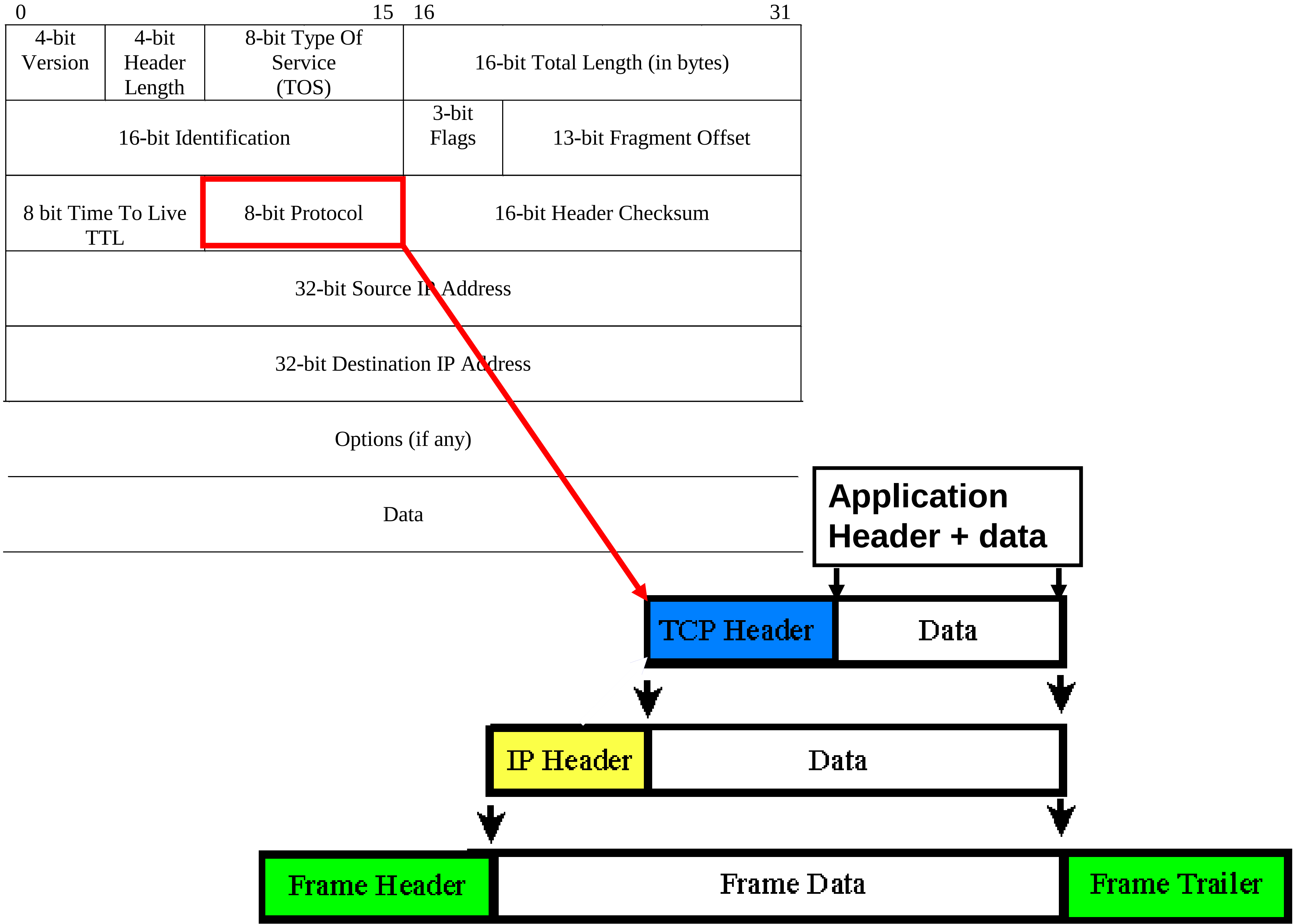
The Protocol Field





- The protocol field determines the Layer 4 protocol being carried within an IP datagram.
- Although much of the IP traffic uses TCP, other protocols can also use UDP, other transport layers, or UDP.
- Each IP header must identify the destination Layer 4 protocol for the datagram.
- Transport layer protocols are numbered, similarly to **port numbers**.
- IP includes the protocol number in the protocol field.

IP Header



<table>
<tr><td colspan="2">0</td><td>15</td><td colspan="3">16   31</td></tr>
<tr><td>4-bit Version</td><td>4-bit Header Length</td><td>8-bit Type Of Service (TOS)</td><td colspan="3">16-bit Total Length (in bytes)</td></tr>
<tr><td colspan="3">16-bit Identification</td><td>3-bit Flags</td><td colspan="2">13-bit Fragment Offset</td></tr>
<tr><td colspan="2">8 bit Time To Live TTL</td><td>8-bit Protocol</td><td colspan="3">16-bit Header Checksum</td></tr>
<tr><td colspan="6">32-bit Source IP Address</td></tr>
<tr><td colspan="6">32-bit Destination IP Address</td></tr>
<tr><td colspan="6">Options (if any)</td></tr>
<tr><td colspan="6">Data</td></tr>
</table>

Application Header + data

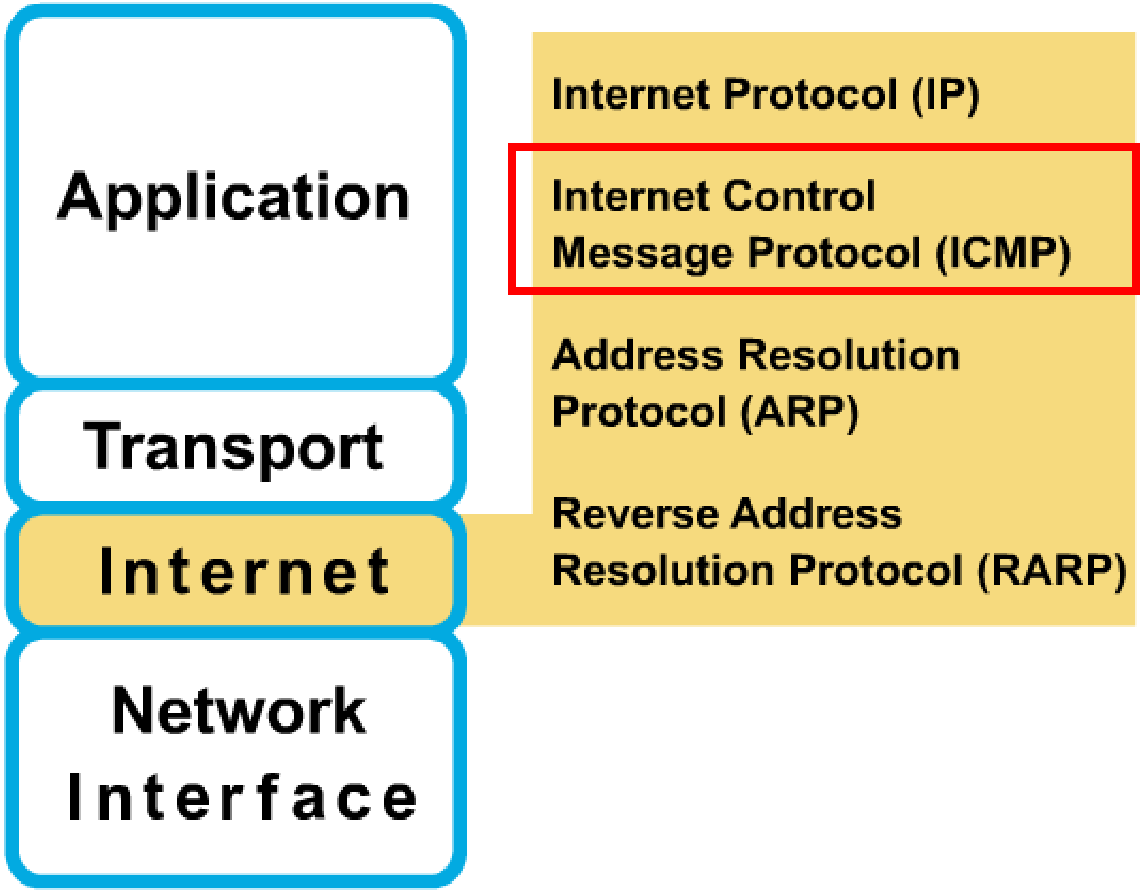
TCP Header | Data

IP Header | Data

Frame Header | Frame Data | Frame Trailer

ICMP – Internet Control Message Protocol

Network Layer Overview



Application

Transport

Internet

Network Interface

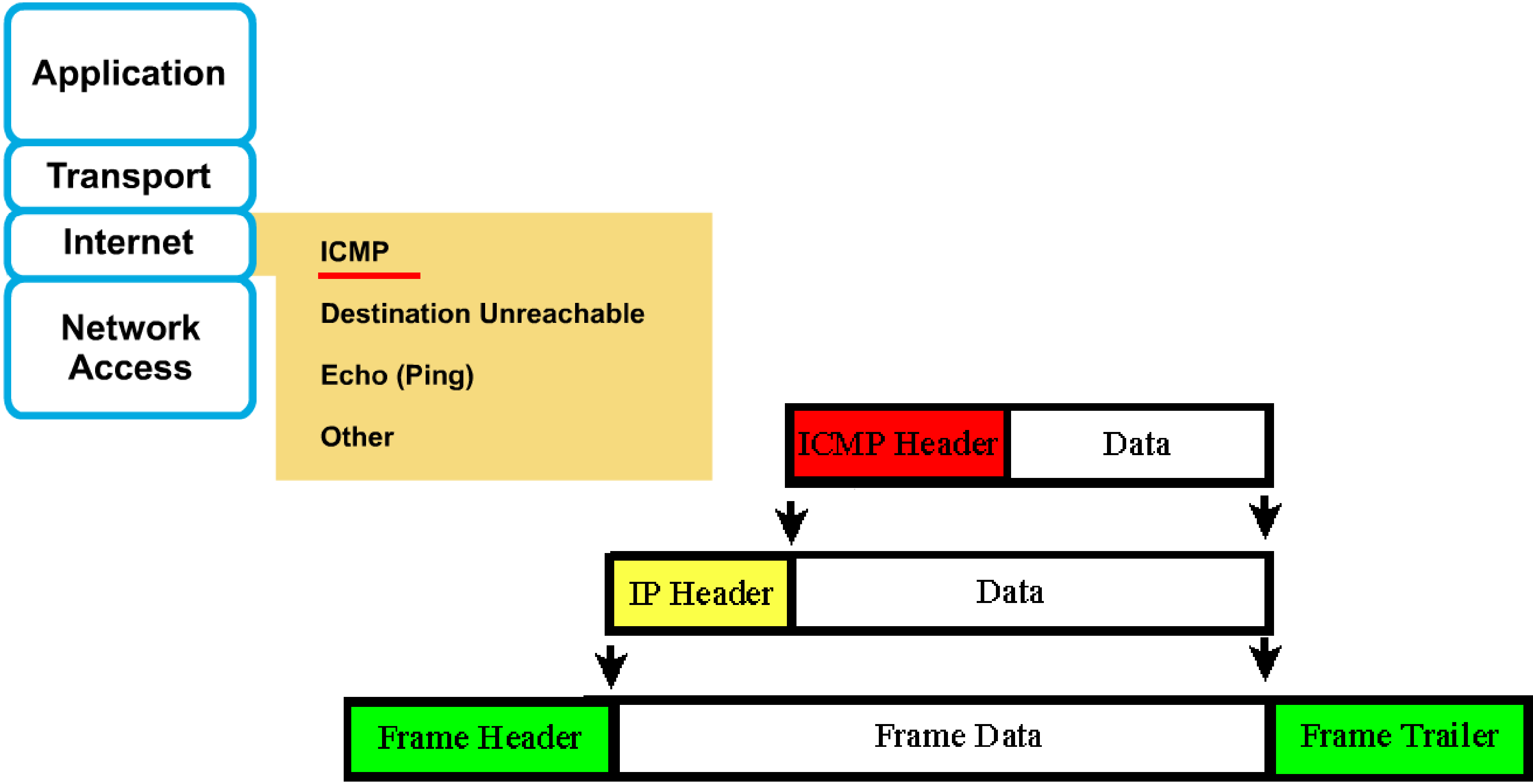
Internet Protocol (IP)

Internet Control Message Protocol (ICMP)

Address Resolution Protocol (ARP)

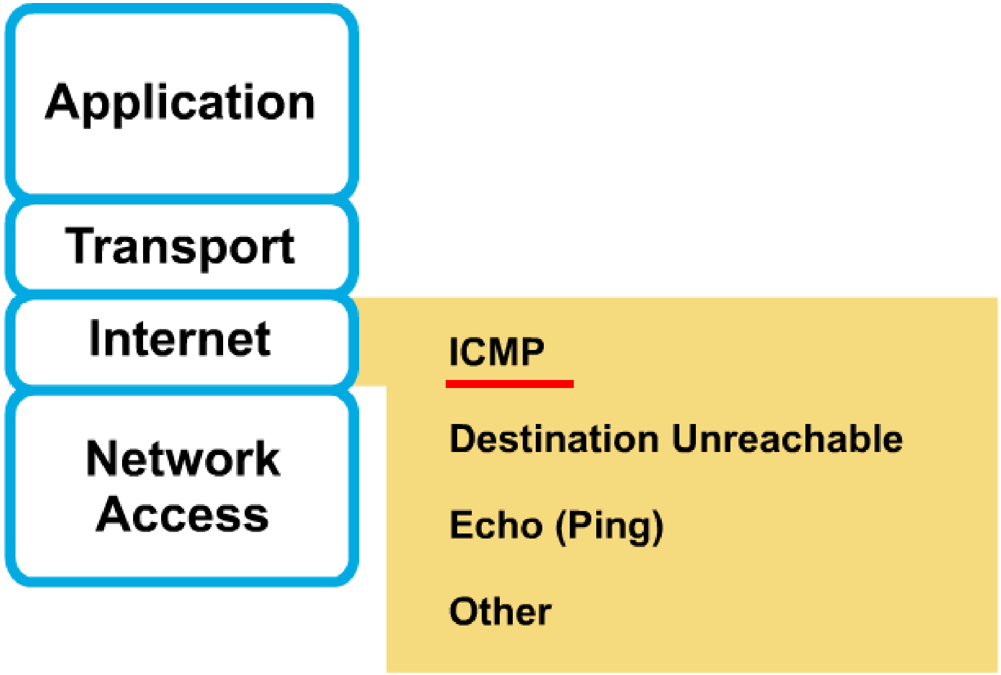
Reverse Address Resolution Protocol (RARP)

Internet Control Message Protocol



- All TCP/IP hosts implement ICMP. ICMP messages are carried in IP datagrams and are used to send error and control messages.

Internet Control Message Protocol



Application

Transport

Internet

Network Access

ICMP

Destination Unreachable

Echo (Ping)

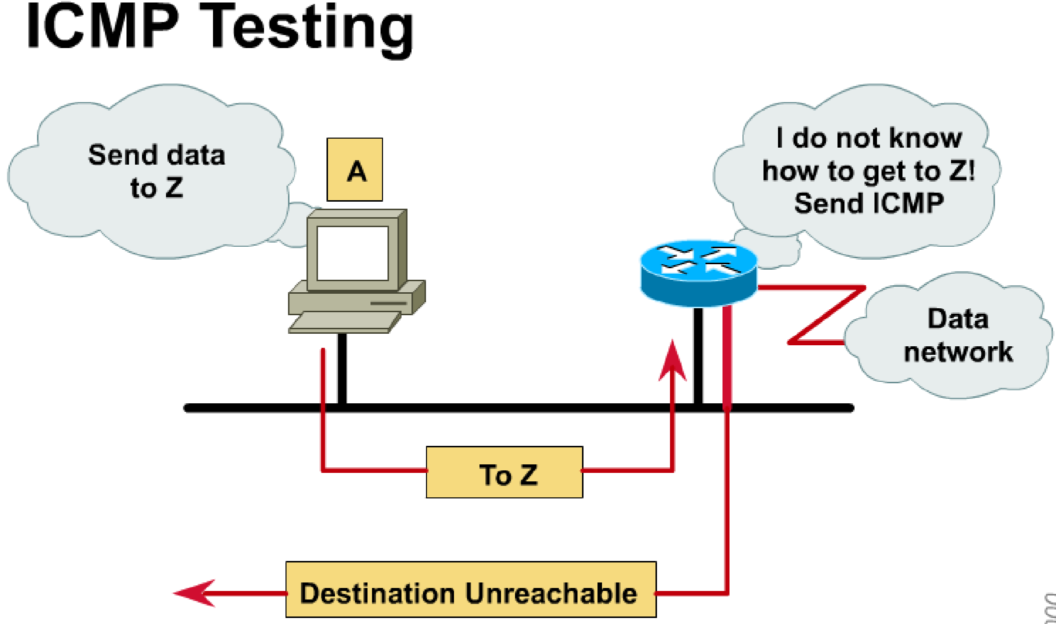
Other

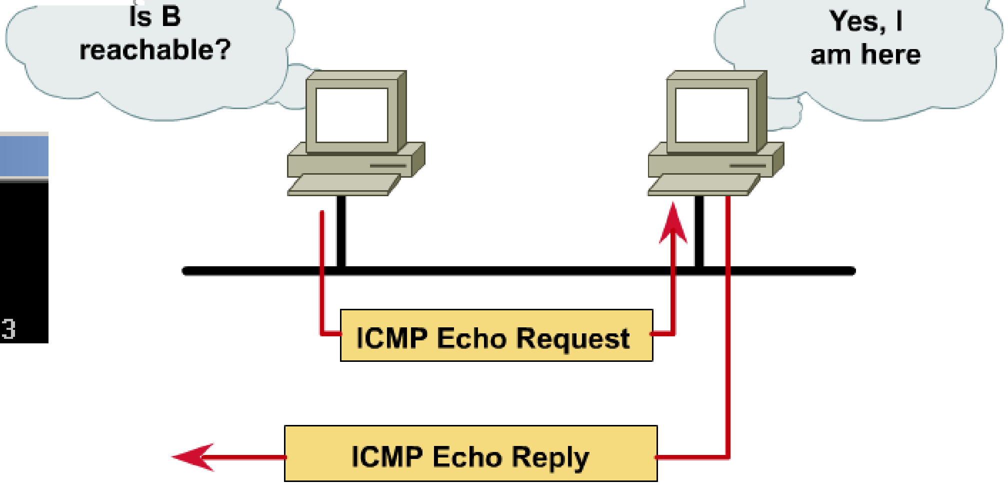
ICMP uses the following types of defined messages.

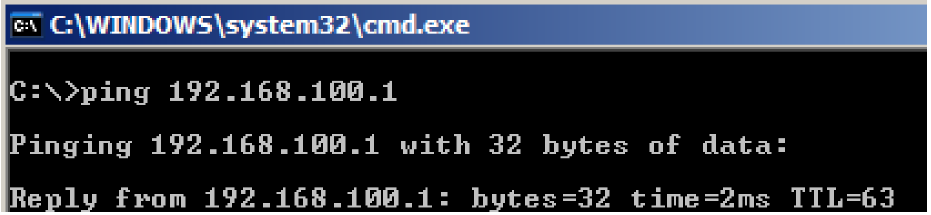
- Destination Unreachable
- Time to Live Exceeded
- Parameter Problem
- Source Quench
- Redirect

- Echo
- Echo Reply
- Timestamp
- Timestamp Reply
- Information Request
- Information Reply
- Address Request
- Address Reply

Ping: ICMP Echo Request and Echo Reply



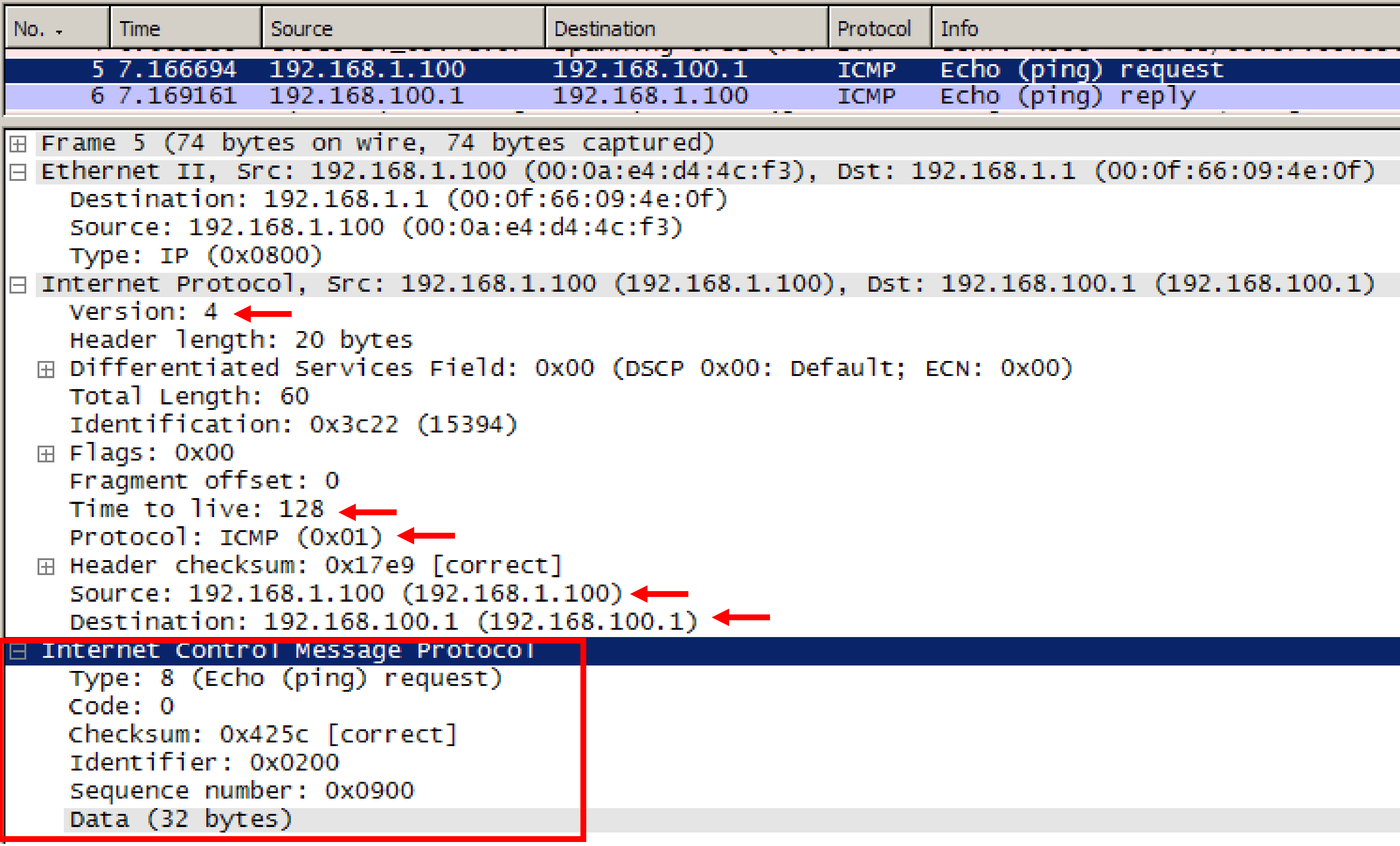




```
C:\>ping 192.168.100.1

Pinging 192.168.100.1 with 32 bytes of data:

Reply from 192.168.100.1: bytes=32 time=2ms TTL=63
```

- We will discuss ping, echo request and echo reply, in detail in the presentation ICMP – Understanding Ping and Traceroute.

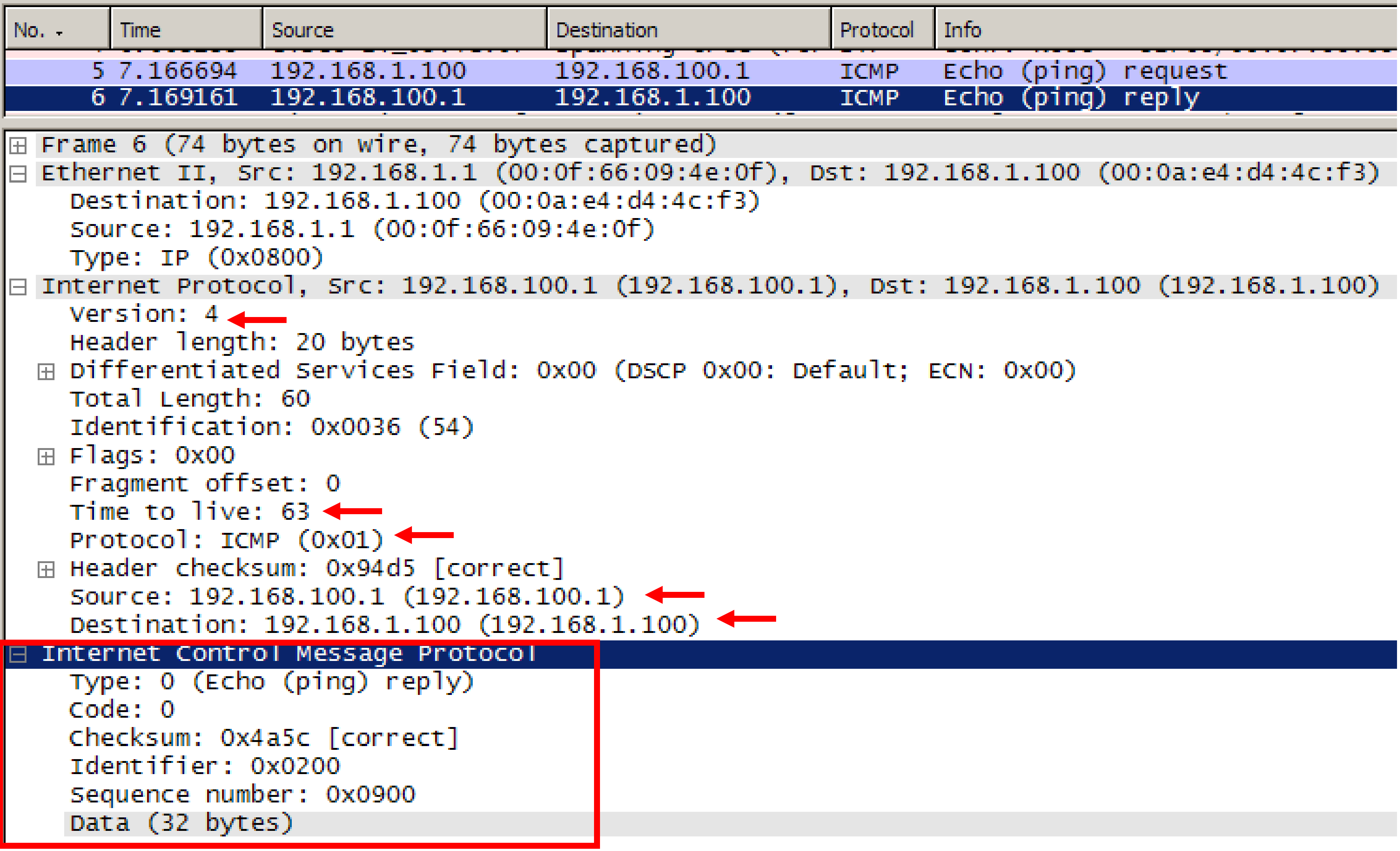
ICMP Echo Request (ping)



```
No. -   Time        Source            Destination       Protocol  Info
   5 7.166694   192.168.1.100     192.168.100.1     ICMP      Echo (ping) request
   6 7.169161   192.168.100.1     192.168.1.100     ICMP      Echo (ping) reply
```

```
⊞ Frame 5 (74 bytes on wire, 74 bytes captured)
⊟ Ethernet II, Src: 192.168.1.100 (00:0a:e4:d4:4c:f3), Dst: 192.168.1.1 (00:0f:66:09:4e:0f)
    Destination: 192.168.1.1 (00:0f:66:09:4e:0f)
    Source: 192.168.1.100 (00:0a:e4:d4:4c:f3)
    Type: IP (0x0800)
⊟ Internet Protocol, Src: 192.168.1.100 (192.168.1.100), Dst: 192.168.100.1 (192.168.100.1)
    Version: 4 ←
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 60
    Identification: 0x3c22 (15394)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 128 ←
    Protocol: ICMP (0x01) ←
  ⊞ Header checksum: 0x17e9 [correct]
    Source: 192.168.1.100 (192.168.1.100) ←
    Destination: 192.168.100.1 (192.168.100.1) ←
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x425c [correct]
    Identifier: 0x0200
    Sequence number: 0x0900
    Data (32 bytes)
```

ICMP Echo Reply (ping)



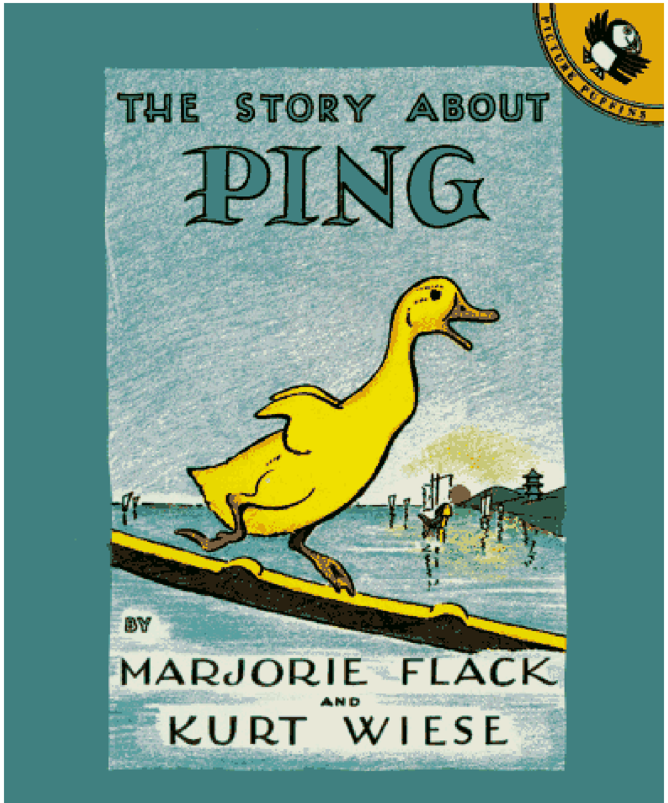
```
No. .  Time      Source          Destination     Protocol  Info
    5 7.166694  192.168.1.100   192.168.100.1   ICMP      Echo (ping) request
    6 7.169161  192.168.100.1   192.168.1.100   ICMP      Echo (ping) reply
```

```
⊞ Frame 6 (74 bytes on wire, 74 bytes captured)
⊟ Ethernet II, Src: 192.168.1.1 (00:0f:66:09:4e:0f), Dst: 192.168.1.100 (00:0a:e4:d4:4c:f3)
    Destination: 192.168.1.100 (00:0a:e4:d4:4c:f3)
    Source: 192.168.1.1 (00:0f:66:09:4e:0f)
    Type: IP (0x0800)
⊟ Internet Protocol, Src: 192.168.100.1 (192.168.100.1), Dst: 192.168.1.100 (192.168.1.100)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 60
    Identification: 0x0036 (54)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 63
    Protocol: ICMP (0x01)
  ⊞ Header checksum: 0x94d5 [correct]
    Source: 192.168.100.1 (192.168.100.1)
    Destination: 192.168.1.100 (192.168.1.100)
⊟ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x4a5c [correct]
    Identifier: 0x0200
    Sequence number: 0x0900
    Data (32 bytes)
```

For more information on Ping

Here are two options for more information on Ping:

- See my PowerPoint presentation: **ICMP – Understanding Ping and Trace**
- Read the book: **The Story About Ping ☺**
by Marjorie Flack, Kurt Wiese (See a Amazon.com customer review on next slide – very funny!



Review of Story of Ping on Amazon.com

8271 of 8518 people found the following review helpful:

Ping! I love that duck!, January 25, 2000

Reviewer: John E. Fracisco (El Segundo, CA USA)

Using deft allegory, the authors have provided an insightful and intuitive explanation of one of Unix's most venerable networking utilities. Even more stunning is that they were clearly working with a very early beta of the program, as their book first appeared in 1933, years (decades!) before the operating system and network infrastructure were finalized.

The book describes networking in terms even a child could understand, choosing to anthropomorphize the underlying packet structure. The ping packet is described as a duck, who, with other packets (more ducks), spends a certain period of time on the host machine (the wise-eyed boat). At the same time each day (I suspect this is scheduled under cron), the little packets (ducks) exit the host (boat) by way of a bridge (a bridge). From the bridge, the packets travel onto the internet (here embodied by the Yangtze River).

The title character -- er, packet, is called Ping. Ping meanders around the river before being received by another host (another boat). He spends a brief time on the other boat, but eventually returns to his original host machine (the wise-eyed boat) somewhat the worse for wear.

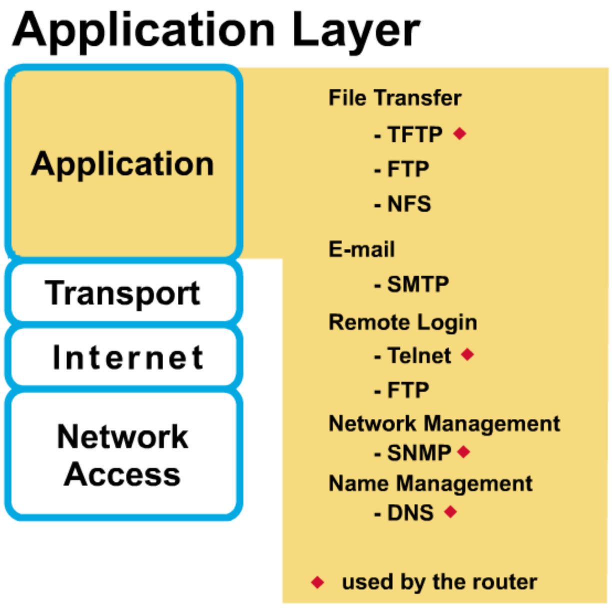
If you need a good, high-level overview of the ping utility, this is the book. I can't recommend it for most managers, as the technical aspects may be too overwhelming and the basic concepts too daunting.

Problems With This Book

As good as it is, The Story About Ping is not without its faults. There is no index, and though the ping(8) man pages cover the command line options well enough, some review of them seems to be in order. Likewise, in a book solely about Ping, I would have expected a more detailed overview of the ICMP packet structure.

But even with these problems, The Story About Ping has earned a place on my bookshelf, right between Stevens' Advanced Programming in the Unix Environment, and my dog-eared copy of Dante's seminal work on MS Windows, Inferno. Who can read that passage on the Windows API ("Obscure, profound it was, and nebulous, So that by fixing on its depths my sight -- Nothing whatever I discerned therein."), without shaking their head with deep understanding. But I digress. *--This text refers to the School & Library Binding edition*.

Ping – A TCP/IP Application



Application Layer

Application

Transport

Internet

Network Access

File Transfer
- TFTP ◆
- FTP
- NFS

E-mail
- SMTP

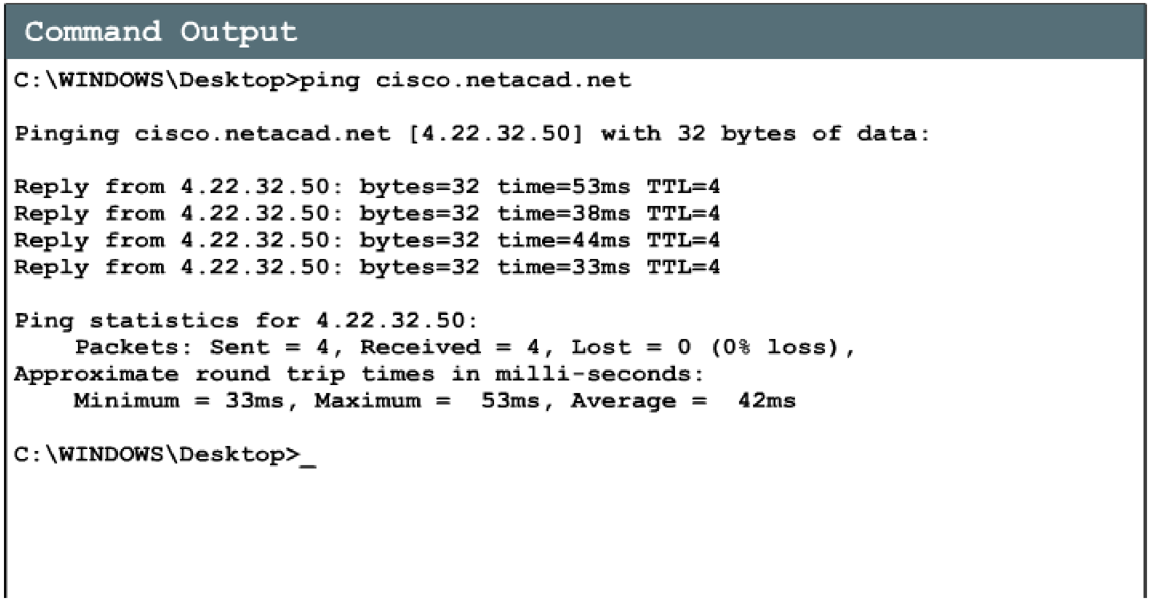
Remote Login
- Telnet ◆
- FTP

Network Management
- SNMP ◆

Name Management
- DNS ◆

◆ used by the router

ping



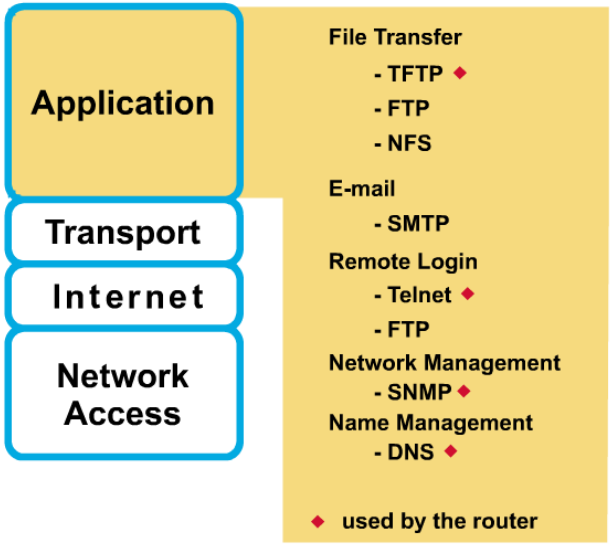
Command Output

```
C:\WINDOWS\Desktop>ping cisco.netacad.net

Pinging cisco.netacad.net [4.22.32.50] with 32 bytes of data:

Reply from 4.22.32.50: bytes=32 time=53ms TTL=4
Reply from 4.22.32.50: bytes=32 time=38ms TTL=4
Reply from 4.22.32.50: bytes=32 time=44ms TTL=4
Reply from 4.22.32.50: bytes=32 time=33ms TTL=4

Ping statistics for 4.22.32.50:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 33ms, Maximum =  53ms, Average =  42ms

C:\WINDOWS\Desktop>_
```

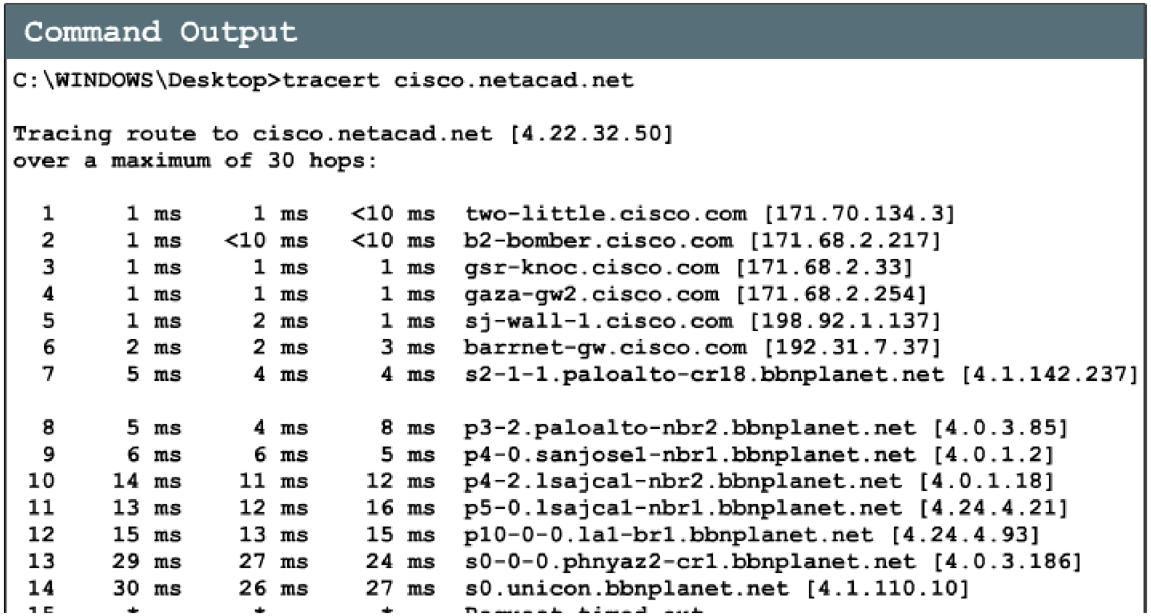
- **PING** (Packet Internet Groper) is a diagnostic utility used to determine whether a computer is properly connected to devices/Internet.
- More in a later presentation!

Traceroute – A TCP/IP Application

Application Layer



tracert

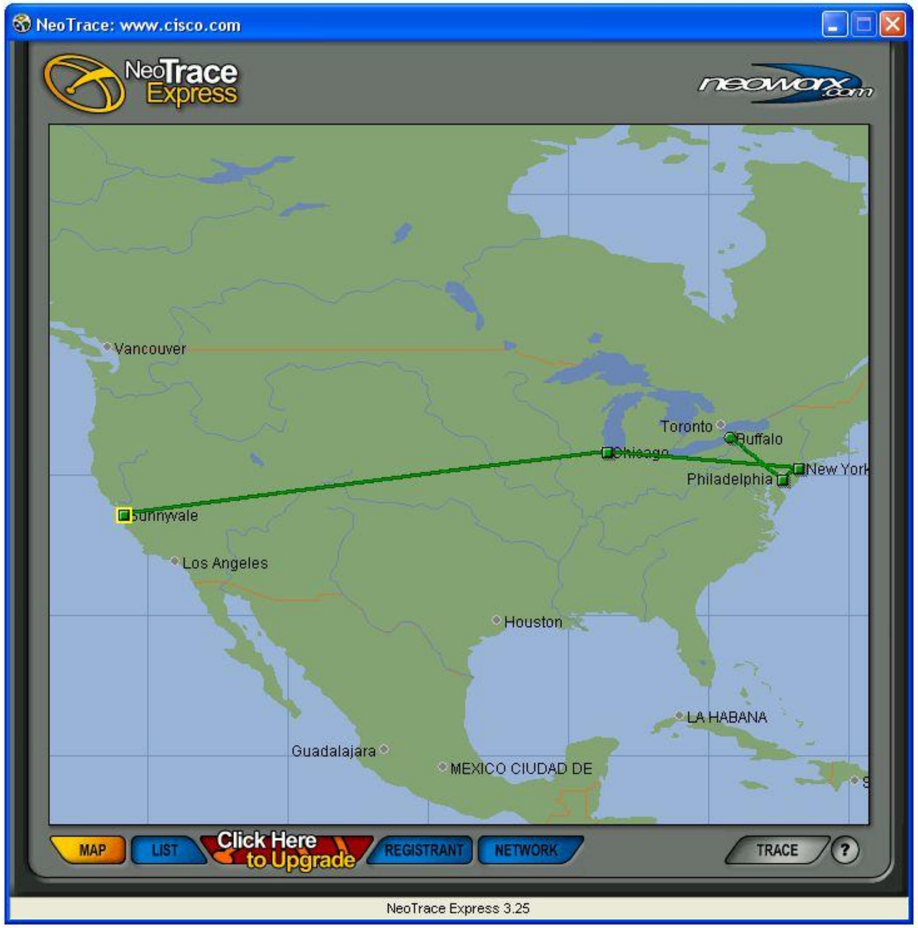


```
C:\WINDOWS\Desktop>tracert cisco.netacad.net

Tracing route to cisco.netacad.net [4.22.32.50]
over a maximum of 30 hops:

  1     1 ms     1 ms   <10 ms  two-little.cisco.com [171.70.134.3]
  2     1 ms   <10 ms   <10 ms  b2-bomber.cisco.com [171.68.2.217]
  3     1 ms     1 ms     1 ms  gsr-knoc.cisco.com [171.68.2.33]
  4     1 ms     1 ms     1 ms  gaza-gw2.cisco.com [171.68.2.254]
  5     1 ms     2 ms     1 ms  sj-wall-1.cisco.com [198.92.1.137]
  6     2 ms     2 ms     3 ms  barrnet-gw.cisco.com [192.31.7.37]
  7     5 ms     4 ms     4 ms  s2-1-1.paloalto-cr18.bbnplanet.net [4.1.142.237]

  8     5 ms     4 ms     8 ms  p3-2.paloalto-nbr2.bbnplanet.net [4.0.3.85]
  9     6 ms     6 ms     5 ms  p4-0.sanjose1-nbr1.bbnplanet.net [4.0.1.2]
 10    14 ms    11 ms    12 ms  p4-2.lsajca1-nbr2.bbnplanet.net [4.0.1.18]
 11    13 ms    12 ms    16 ms  p5-0.lsajca1-nbr1.bbnplanet.net [4.24.4.21]
 12    15 ms    13 ms    15 ms  p10-0-0.la1-br1.bbnplanet.net [4.24.4.93]
 13    29 ms    27 ms    24 ms  s0-0-0.phnyaz2-cr1.bbnplanet.net [4.0.3.186]
 14    30 ms    26 ms    27 ms  s0.unicon.bbnplanet.net [4.1.110.10]
 15     *        *        *     Request timed out.
```

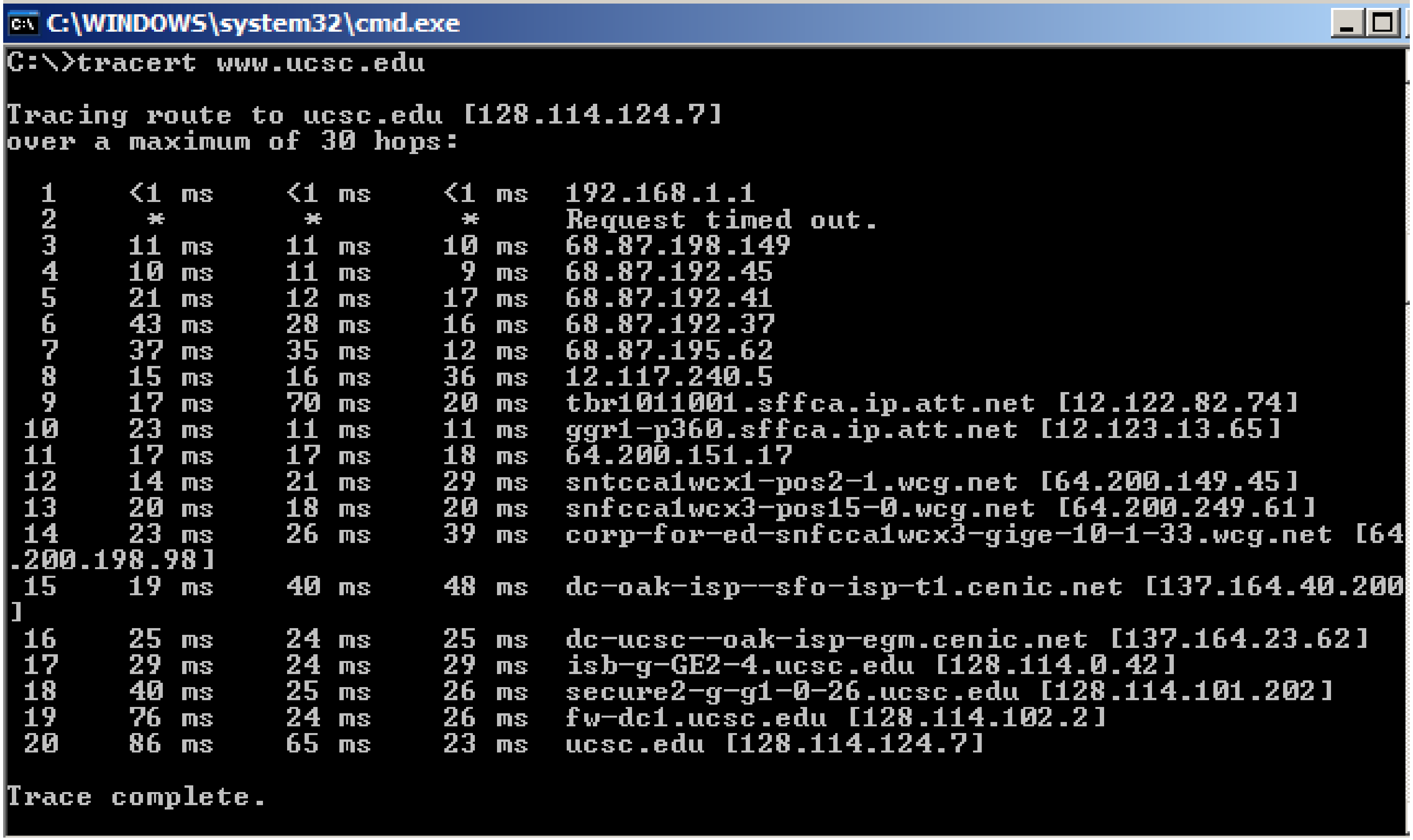
- **Traceroute** is a program that is available on many systems, and is similar to PING, except that traceroute provides more information than PING.
- Traceroute traces the path a packet takes to a destination, and is used to debug routing problems.
- More in a later presentation!

Traceroute – A TCP/IP Application

- Graphical Trace Programs like NeoTrace (now by McAfee)
- http://www.networkingfiles.com/PingFinger/Neotraceexpress.htm

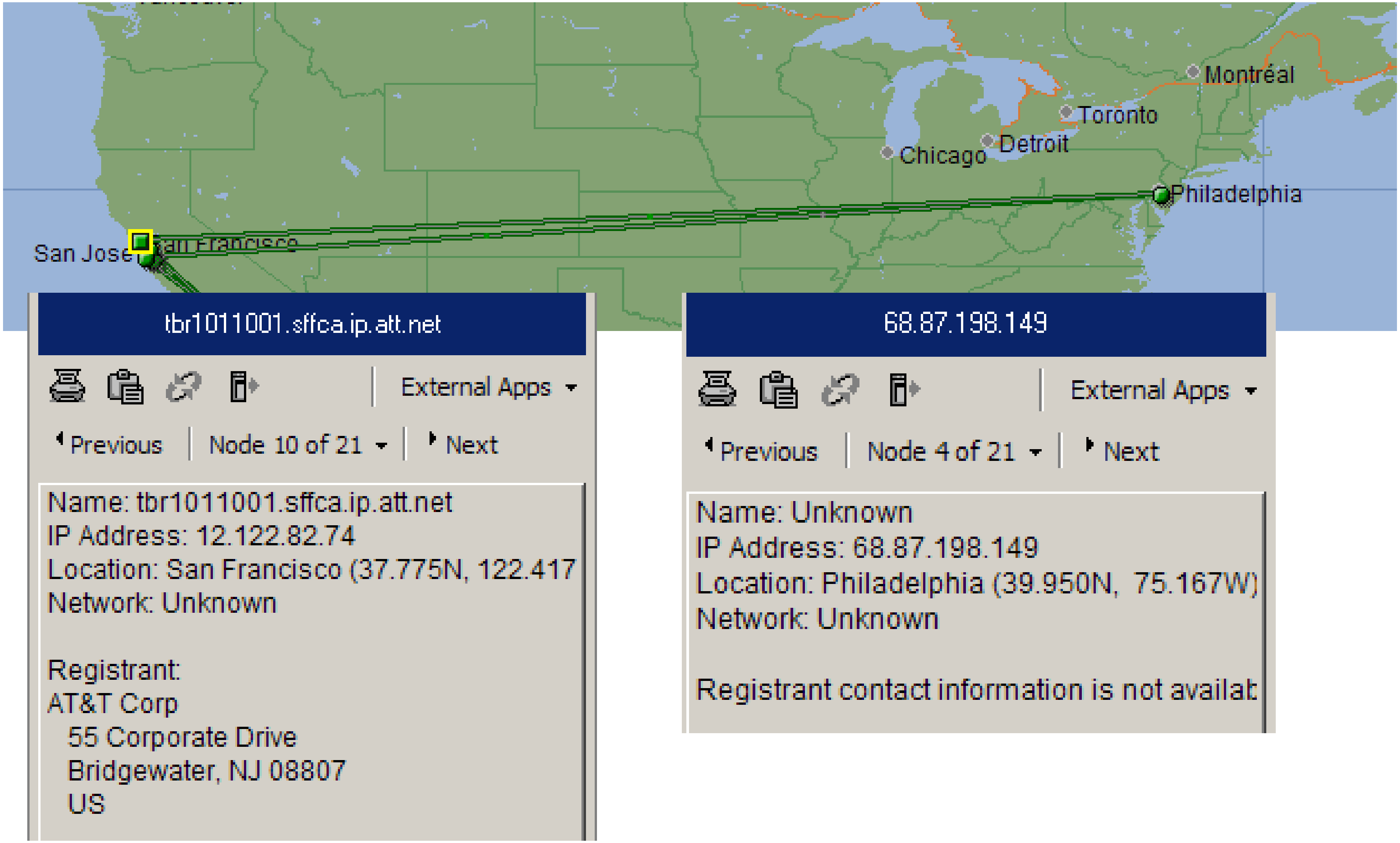


Windows: tracert command



```
C:\>tracert www.ucsc.edu

Tracing route to ucsc.edu [128.114.124.7]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.1.1
  2     *        *        *     Request timed out.
  3    11 ms    11 ms    10 ms  68.87.198.149
  4    10 ms    11 ms     9 ms  68.87.192.45
  5    21 ms    12 ms    17 ms  68.87.192.41
  6    43 ms    28 ms    16 ms  68.87.192.37
  7    37 ms    35 ms    12 ms  68.87.195.62
  8    15 ms    16 ms    36 ms  12.117.240.5
  9    17 ms    70 ms    20 ms  tbr1011001.sffca.ip.att.net [12.122.82.74]
 10    23 ms    11 ms    11 ms  ggr1-p360.sffca.ip.att.net [12.123.13.65]
 11    17 ms    17 ms    18 ms  64.200.151.17
 12    14 ms    21 ms    29 ms  sntcca1wcx1-pos2-1.wcg.net [64.200.149.45]
 13    20 ms    18 ms    20 ms  snfcca1wcx3-pos15-0.wcg.net [64.200.249.61]
 14    23 ms    26 ms    39 ms  corp-for-ed-snfcca1wcx3-gige-10-1-33.wcg.net [64
.200.198.98]
 15    19 ms    40 ms    48 ms  dc-oak-isp--sfo-isp-t1.cenic.net [137.164.40.200
]
 16    25 ms    24 ms    25 ms  dc-ucsc--oak-isp-egm.cenic.net [137.164.23.62]
 17    29 ms    24 ms    29 ms  isb-g-GE2-4.ucsc.edu [128.114.0.42]
 18    40 ms    25 ms    26 ms  secure2-g-g1-0-26.ucsc.edu [128.114.101.202]
 19    76 ms    24 ms    26 ms  fw-dc1.ucsc.edu [128.114.102.2]
 20    86 ms    65 ms    23 ms  ucsc.edu [128.114.124.7]

Trace complete.
```

NeoTrace Map View



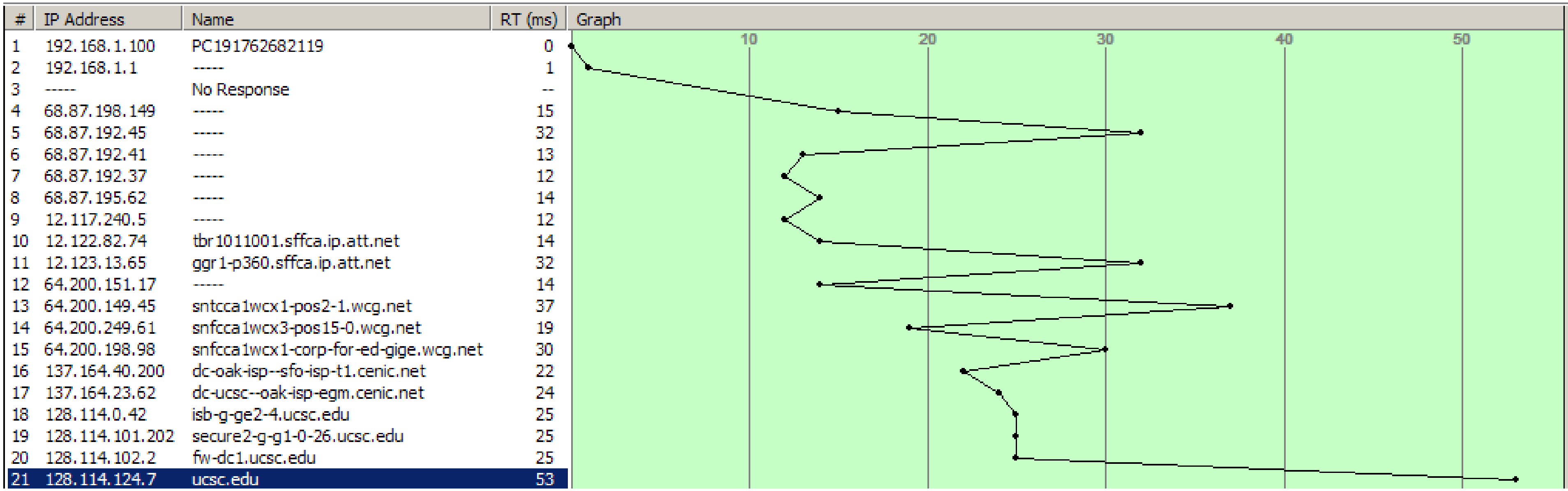
Name: tbr1011001.sffca.ip.att.net
IP Address: 12.122.82.74
Location: San Francisco (37.775N, 122.417
Network: Unknown

Registrant:
AT&T Corp
55 Corporate Drive
Bridgewater, NJ 08807
US

Name: Unknown
IP Address: 68.87.198.149
Location: Philadelphia (39.950N, 75.167W)
Network: Unknown

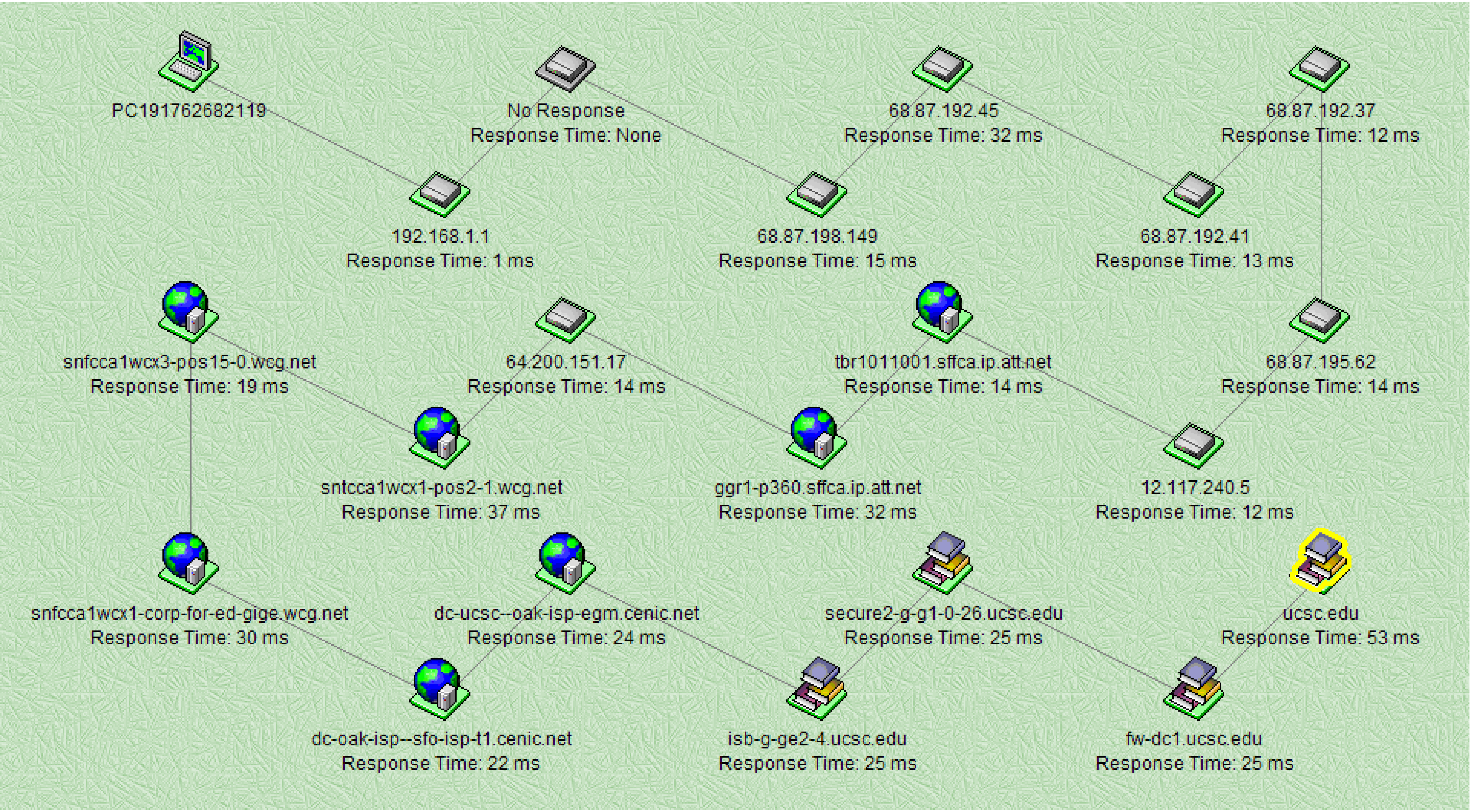
Registrant contact information is not availat

NeoTrace List View

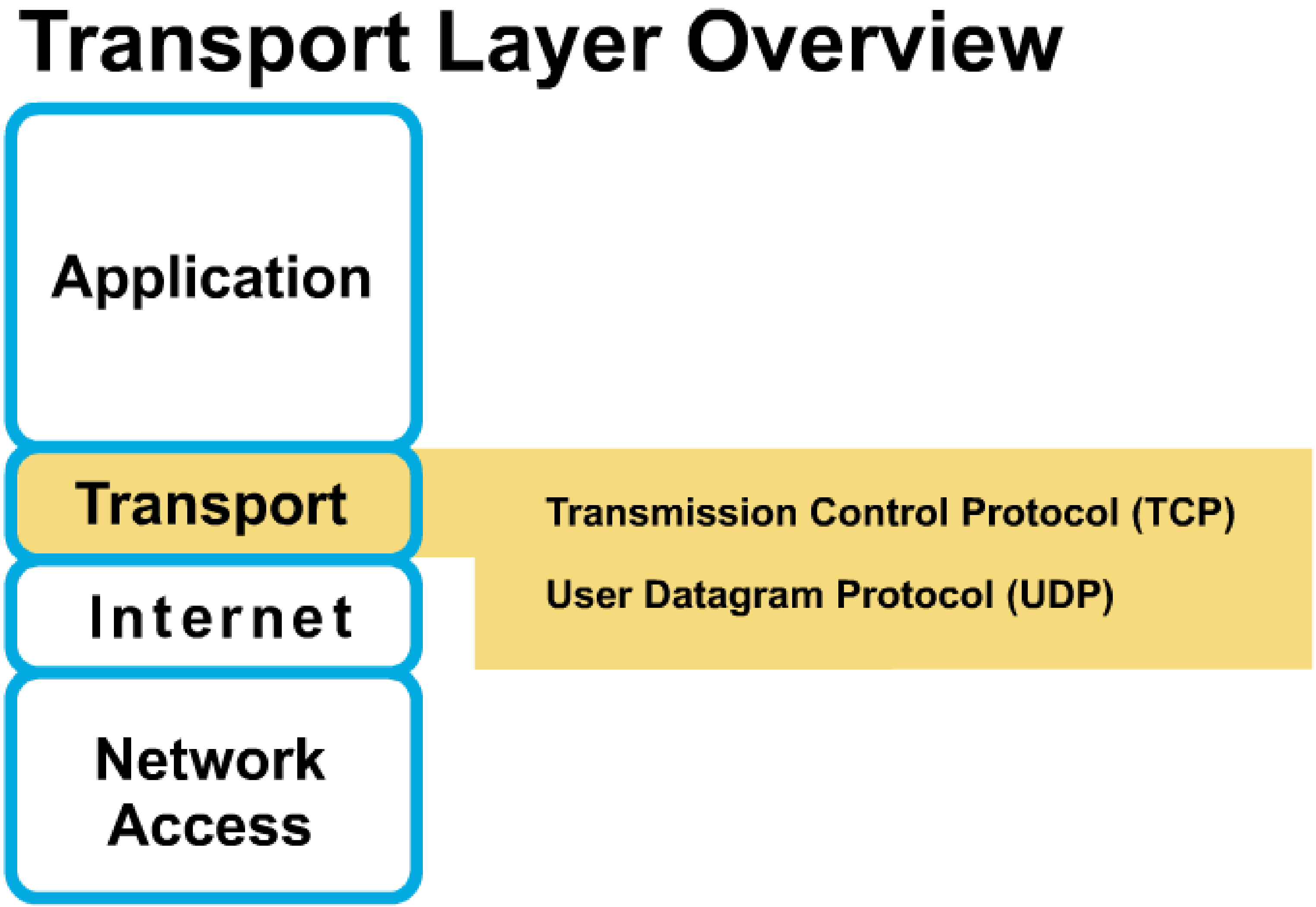


#	IP Address	Name	RT (ms)
1	192.168.1.100	PC191762682119	0
2	192.168.1.1	-----	1
3	-----	No Response	--
4	68.87.198.149	-----	15
5	68.87.192.45	-----	32
6	68.87.192.41	-----	13
7	68.87.192.37	-----	12
8	68.87.195.62	-----	14
9	12.117.240.5	-----	12
10	12.122.82.74	tbr1011001.sffca.ip.att.net	14
11	12.123.13.65	ggr1-p360.sffca.ip.att.net	32
12	64.200.151.17	-----	14
13	64.200.149.45	sntcca1wcx1-pos2-1.wcg.net	37
14	64.200.249.61	snfcca1wcx3-pos15-0.wcg.net	19
15	64.200.198.98	snfcca1wcx1-corp-for-ed-gige.wcg.net	30
16	137.164.40.200	dc-oak-isp--sfo-isp-t1.cenic.net	22
17	137.164.23.62	dc-ucsc--oak-isp-egm.cenic.net	24
18	128.114.0.42	isb-g-ge2-4.ucsc.edu	25
19	128.114.101.202	secure2-g-g1-0-26.ucsc.edu	25
20	128.114.102.2	fw-dc1.ucsc.edu	25
21	128.114.124.7	ucsc.edu	53

NeoTrace Node View



Layer 4: TCP/IP Transport Layer



Transport Layer Overview

Application

Transport

Internet

Network Access

Transmission Control Protocol (TCP)

User Datagram Protocol (UDP)

Topics

Layer 3 Concepts

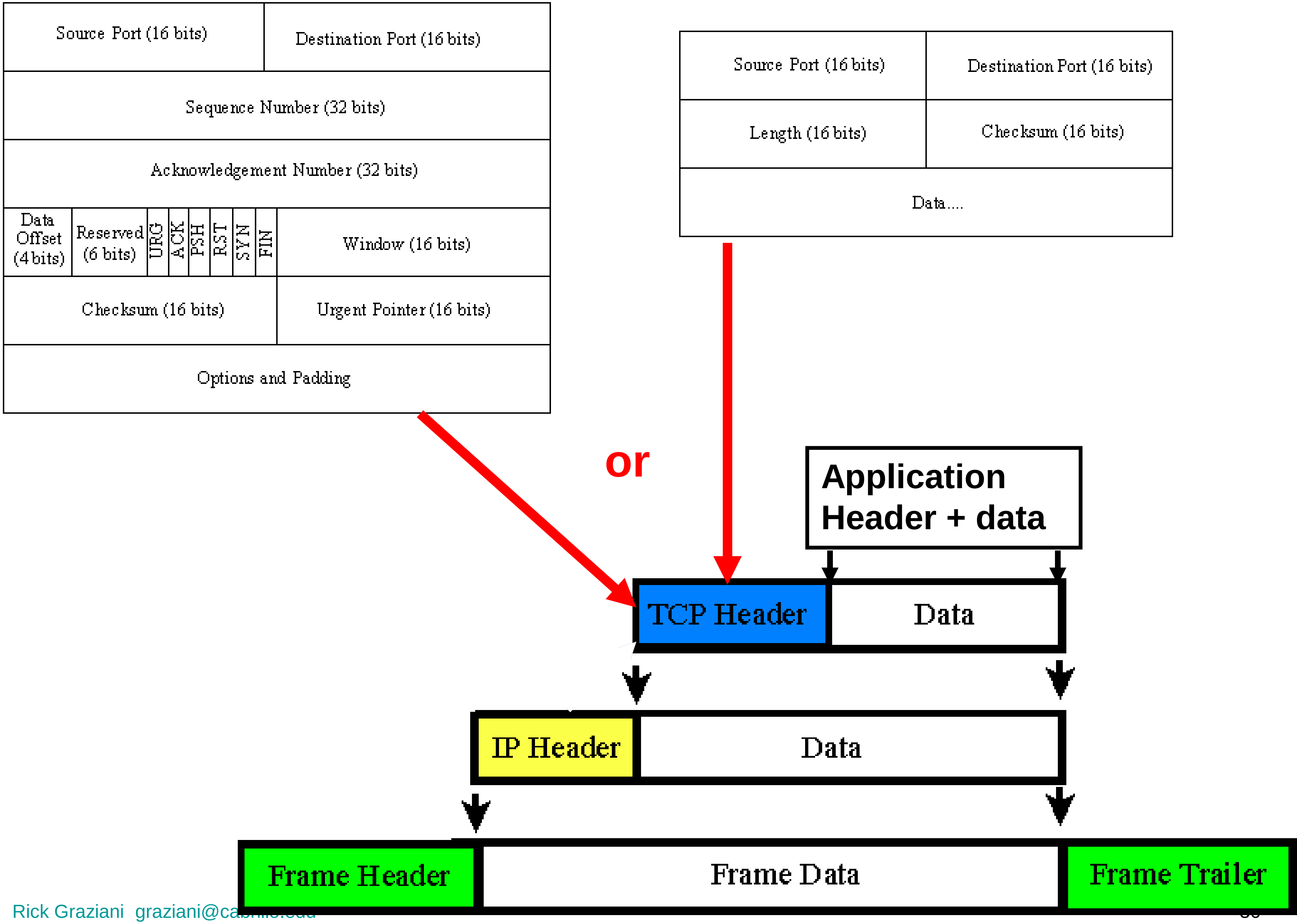
- TCP/IP and the Internet Layer
- Diagram the IP datagram
- Internet Control Message Protocol (ICMP)

TCP/IP protocol stack and the transport layer

- TCP and UDP segment format
- TCP and UDP port numbers
- TCP three-way handshake/open connection
- TCP simple acknowledgment and windowing

TCP Header

UDP Header



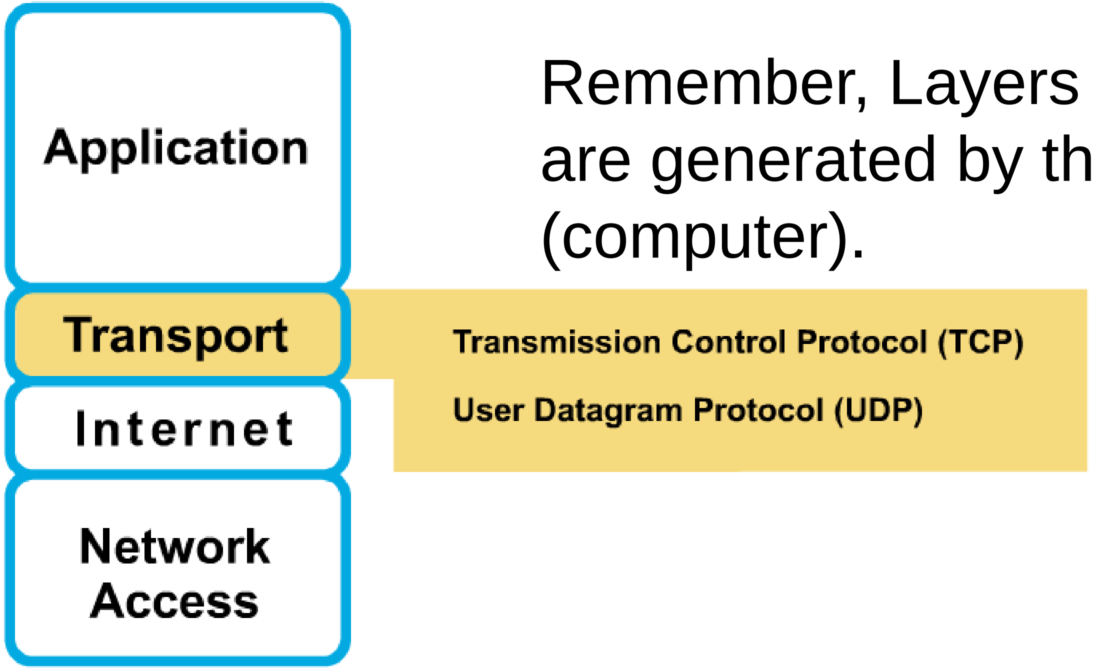
Source Port (16 bits)
Destination Port (16 bits)
Sequence Number (32 bits)
Acknowledgement Number (32 bits)
Data Offset (4 bits)
Reserved (6 bits)
URG
ACK
PSH
RST
SYN
FIN
Window (16 bits)
Checksum (16 bits)
Urgent Pointer (16 bits)
Options and Padding

Source Port (16 bits)
Destination Port (16 bits)
Length (16 bits)
Checksum (16 bits)
Data....

or

Application Header + data

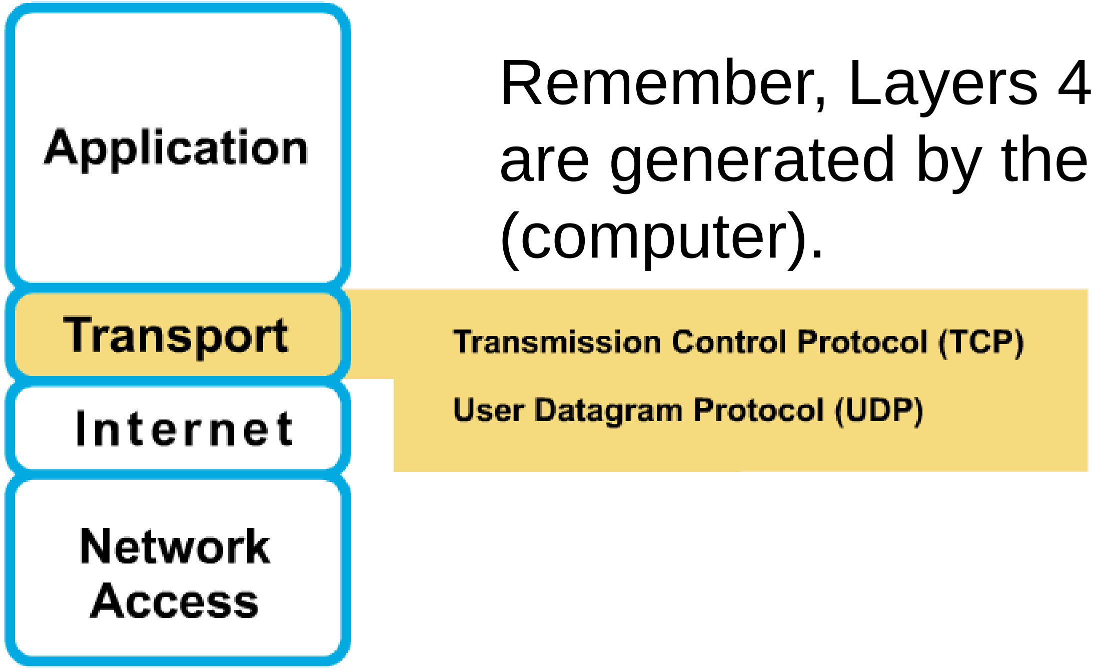
Transport Layer Overview


Application
Transport
Internet
Network Access
Transmission Control Protocol (TCP)
User Datagram Protocol (UDP)

Remember, Layers 4 and above are generated by the host device (computer).

- The transport layer enables a user's device to **segment** several upper-layer applications for placement on the same Layer 4 data stream, and enables a receiving device to reassemble the upper-layer application segments.
- The Layer 4 data stream is a logical connection between the endpoints of a network, and provides transport services from a host to a destination.
- This service is sometimes referred to as **end-to-end service**.

Transport Layer Overview


Application
Transport
Internet
Network Access
Transmission Control Protocol (TCP)
User Datagram Protocol (UDP)

Remember, Layers 4 and above are generated by the host device (computer).

The transport layer also provides two protocols

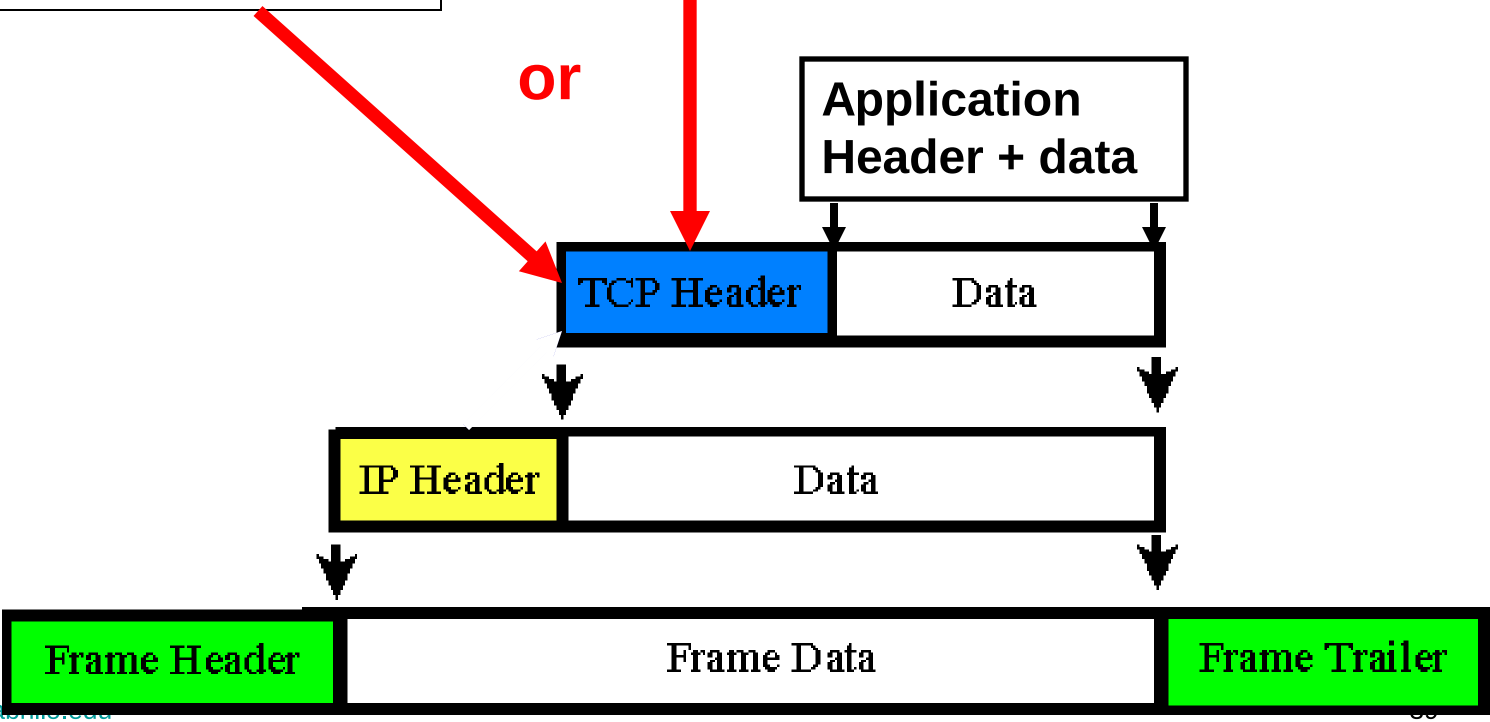
- **TCP** – Transmission Control Protocol
- **UDP** – User Datagram Protocol

TCP Header

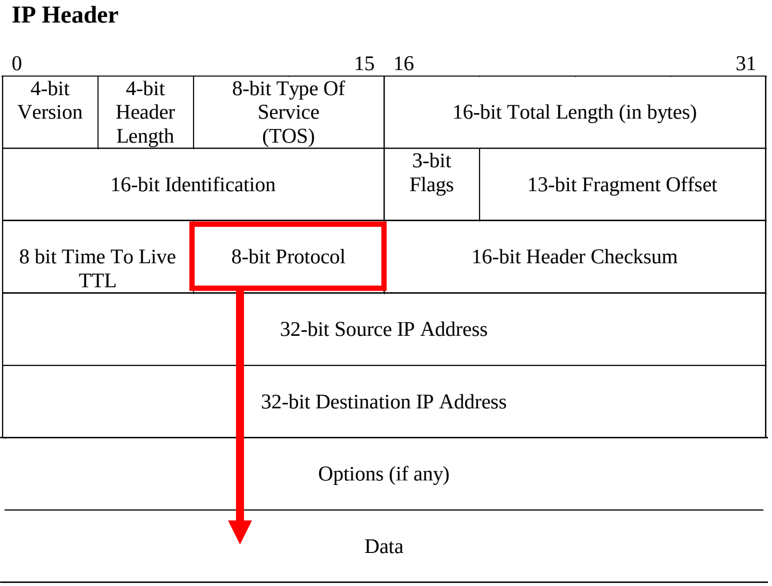
<table>
<tr><td colspan="4">Source Port (16 bits)</td><td colspan="5">Destination Port (16 bits)</td></tr>
<tr><td colspan="9">Sequence Number (32 bits)</td></tr>
<tr><td colspan="9">Acknowledgement Number (32 bits)</td></tr>
<tr><td>Data Offset (4 bits)</td><td>Reserved (6 bits)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Window (16 bits)</td></tr>
<tr><td colspan="8">Checksum (16 bits)</td><td>Urgent Pointer (16 bits)</td></tr>
<tr><td colspan="9">Options and Padding</td></tr>
</table>

UDP Header

Source Port (16 bits)	Destination Port (16 bits)
Length (16 bits)	Checksum (16 bits)
Data....	

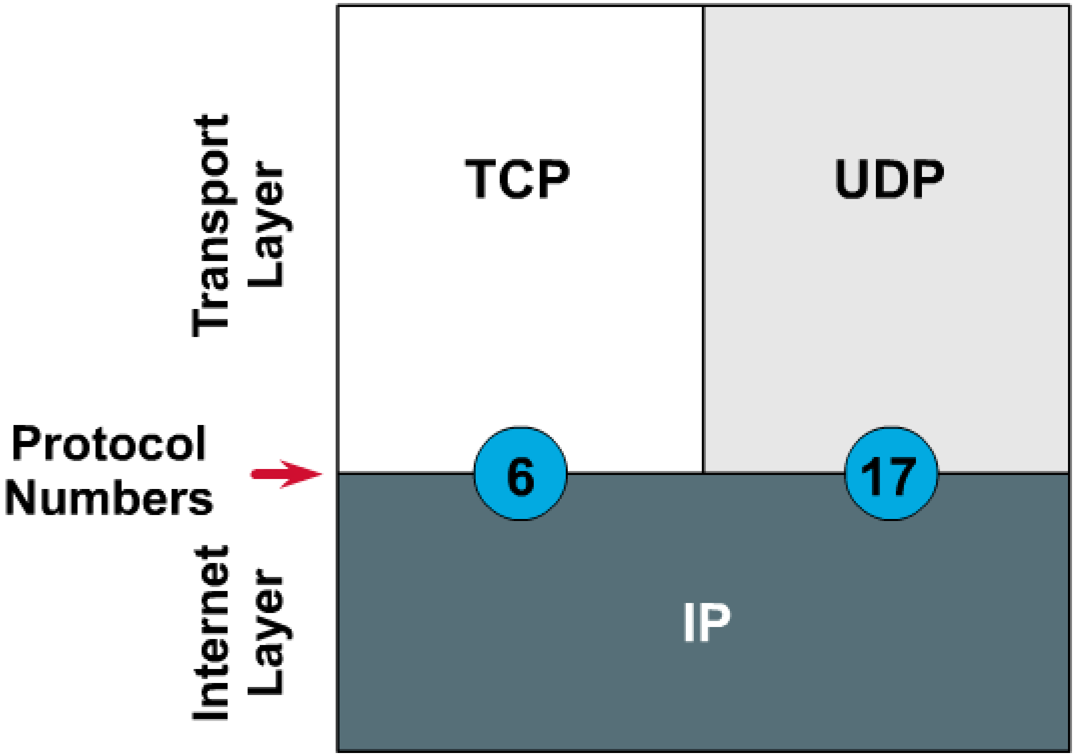


The Protocol Field

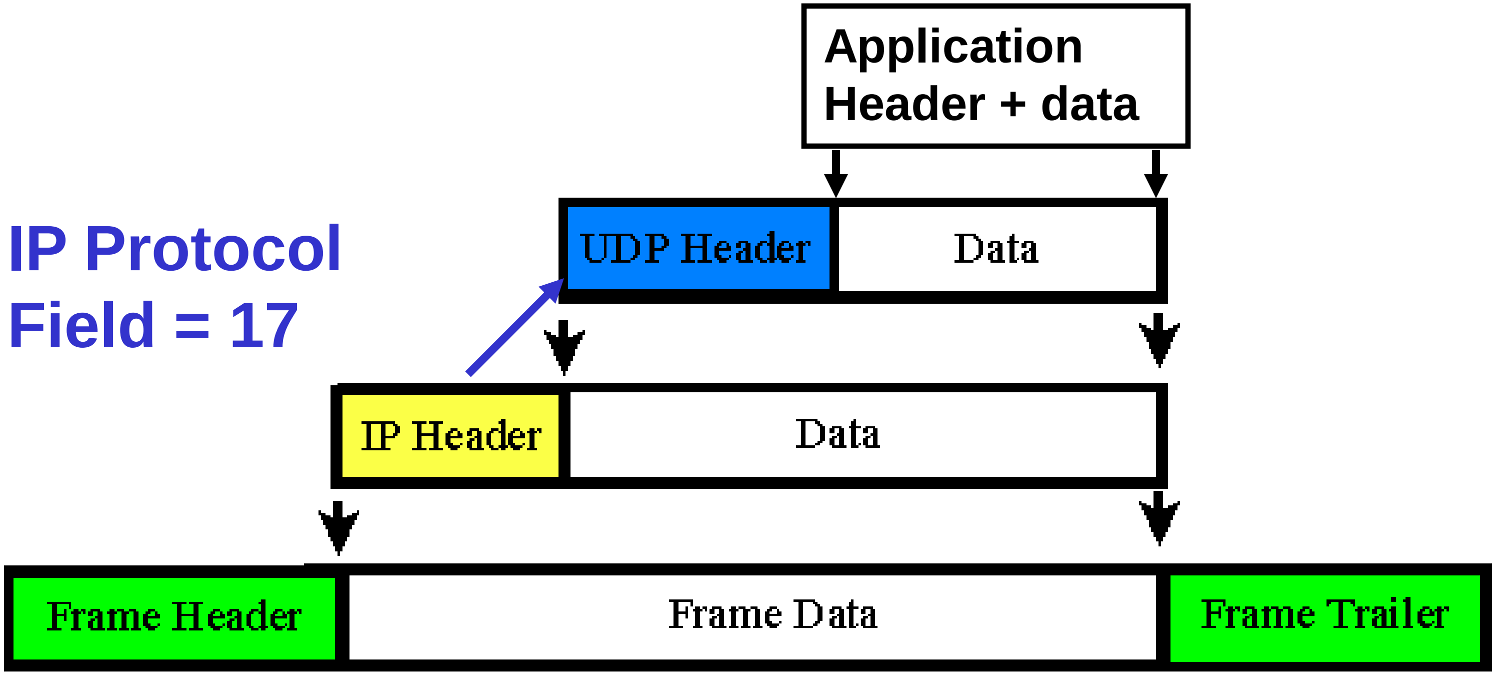


IP Header

0		15	16		31
4-bit Version	4-bit Header Length	8-bit Type Of Service (TOS)	16-bit Total Length (in bytes)		
16-bit Identification			3-bit Flags	13-bit Fragment Offset	
8 bit Time To Live TTL		8-bit Protocol	16-bit Header Checksum		
32-bit Source IP Address					
32-bit Destination IP Address					
Options (if any)					
Data					



IP Packet has a **Protocol field** that specifies whether the segment is TCP or UDP.



IP Protocol Field = 17

Application Header + data

UDP Header

Data

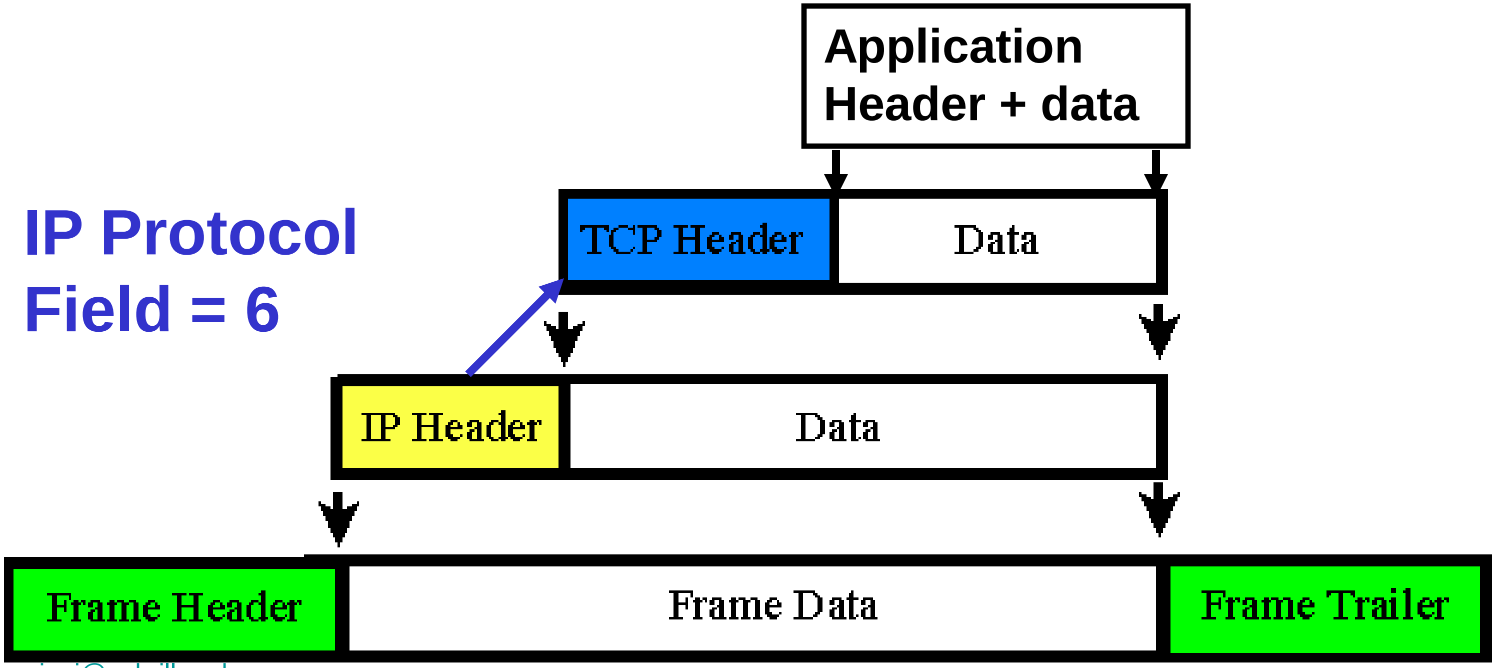
IP Header

Data

Frame Header

Frame Data

Frame Trailer



IP Protocol Field = 6

Application Header + data

TCP Header

Data

IP Header

Data

Frame Header

Frame Data

Frame Trailer

Topics

Layer 3 Concepts

- TCP/IP and the Internet Layer
- Diagram the IP datagram
- Internet Control Message Protocol (ICMP)

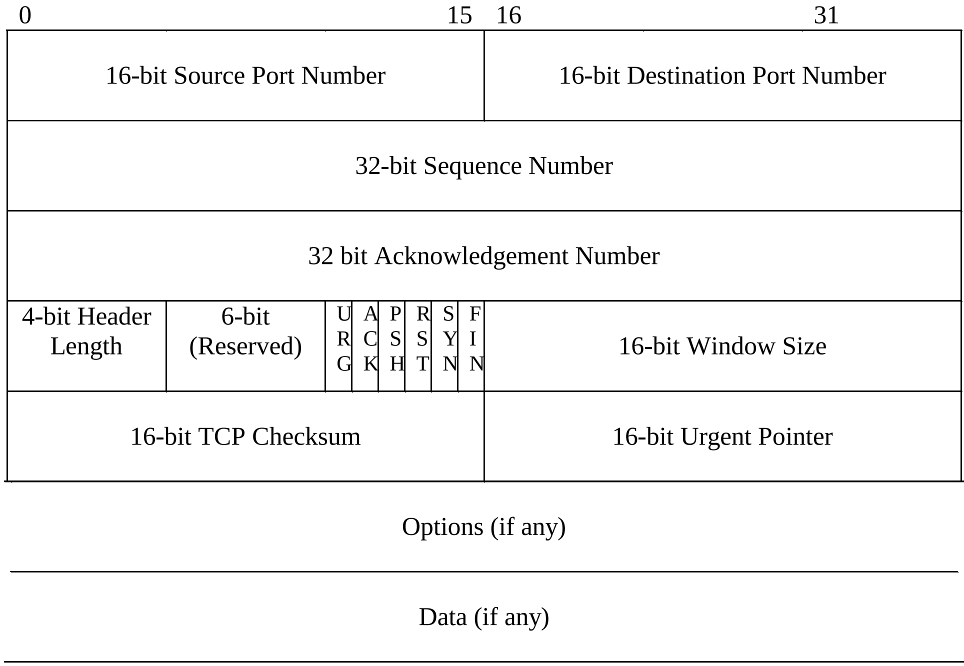
TCP/IP protocol stack and the transport layer

- TCP and UDP segment format
- TCP and UDP port numbers
- TCP three-way handshake/open connection
- TCP simple acknowledgment and windowing

TCP Segment Header

<table>
<tr><td colspan="2">0</td><td colspan="6">15</td><td>16 31</td></tr>
<tr><td colspan="8">16-bit Source Port Number</td><td>16-bit Destination Port Number</td></tr>
<tr><td colspan="9">32-bit Sequence Number</td></tr>
<tr><td colspan="9">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td>16-bit Window Size</td></tr>
<tr><td colspan="8">16-bit TCP Checksum</td><td>16-bit Urgent Pointer</td></tr>
<tr><td colspan="9">Options (if any)</td></tr>
<tr><td colspan="9">Data (if any)</td></tr>
</table>

TCP Segment Header



0 15 16 31

16-bit Source Port Number

16-bit Destination Port Number

32-bit Sequence Number

32 bit Acknowledgement Number

4-bit Header Length

6-bit (Reserved)

URG ACK PSH RST SYN FIN

16-bit Window Size

16-bit TCP Checksum

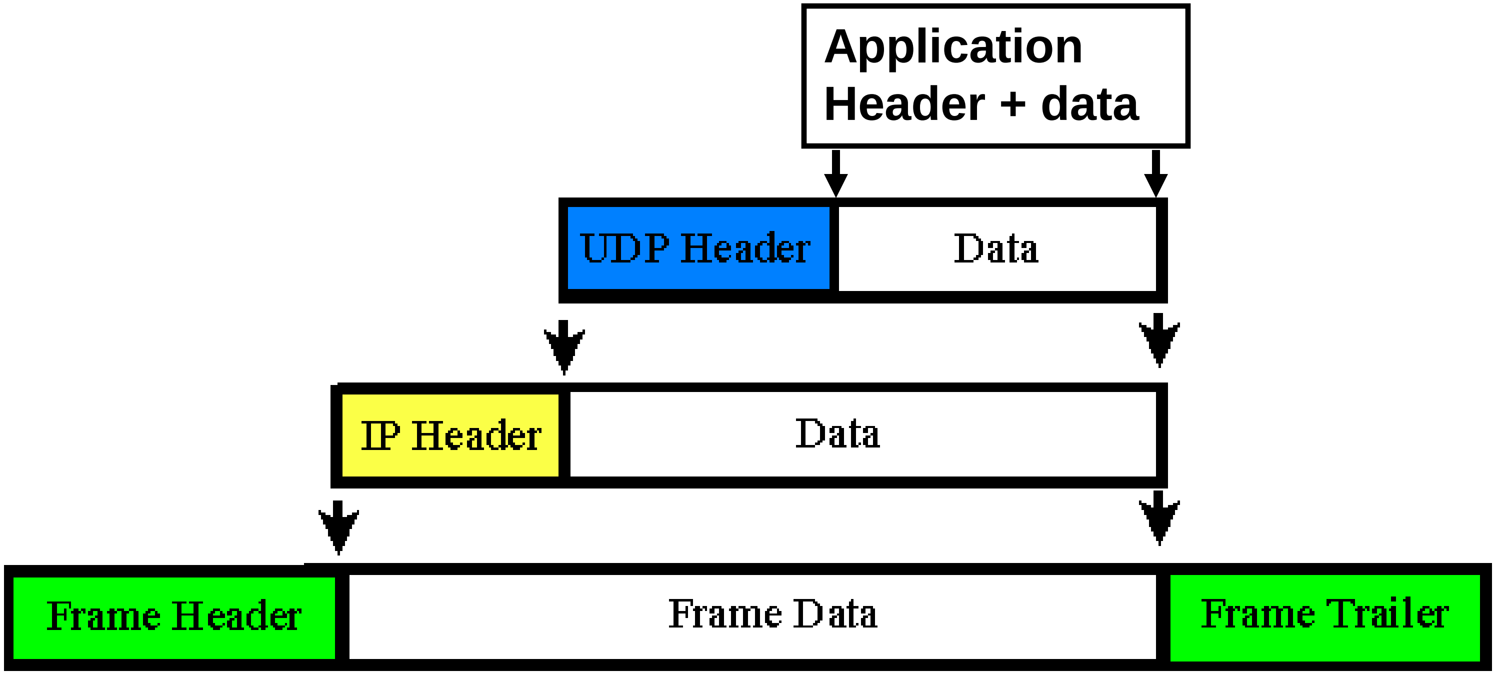
16-bit Urgent Pointer

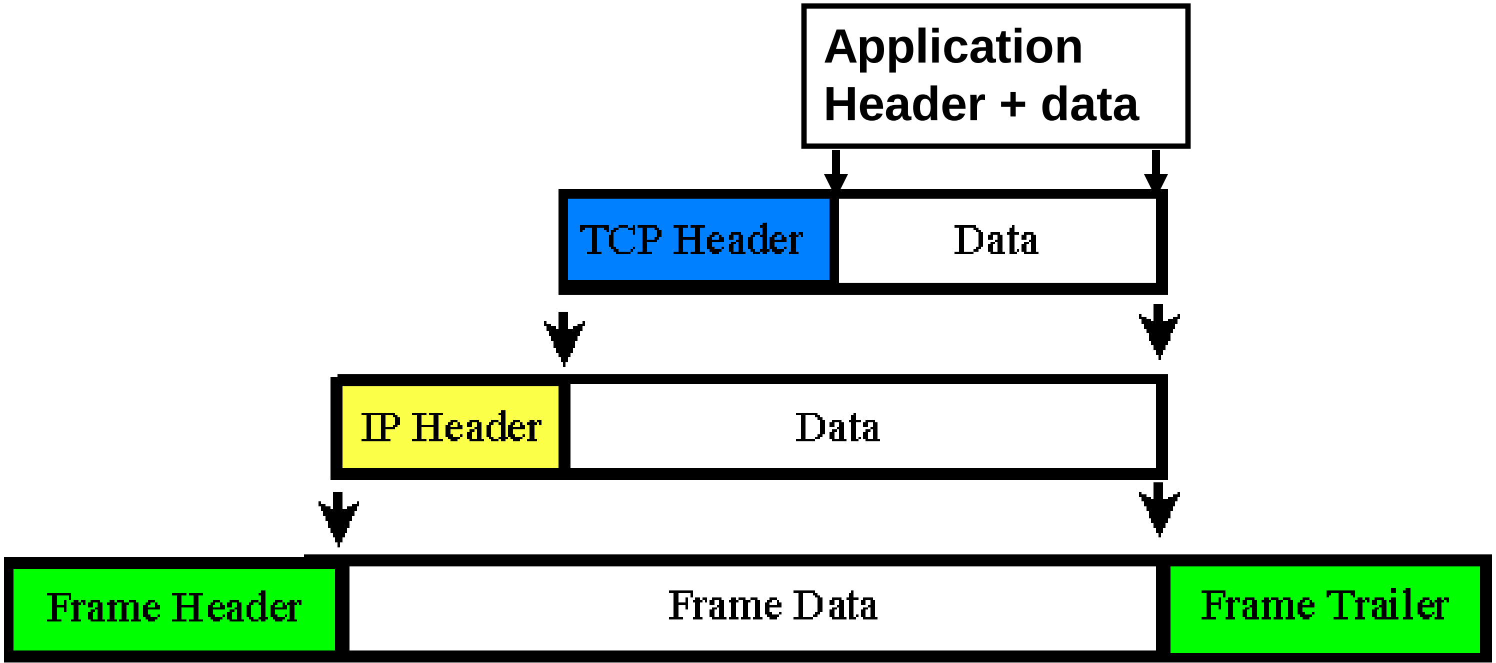
Options (if any)

Data (if any)

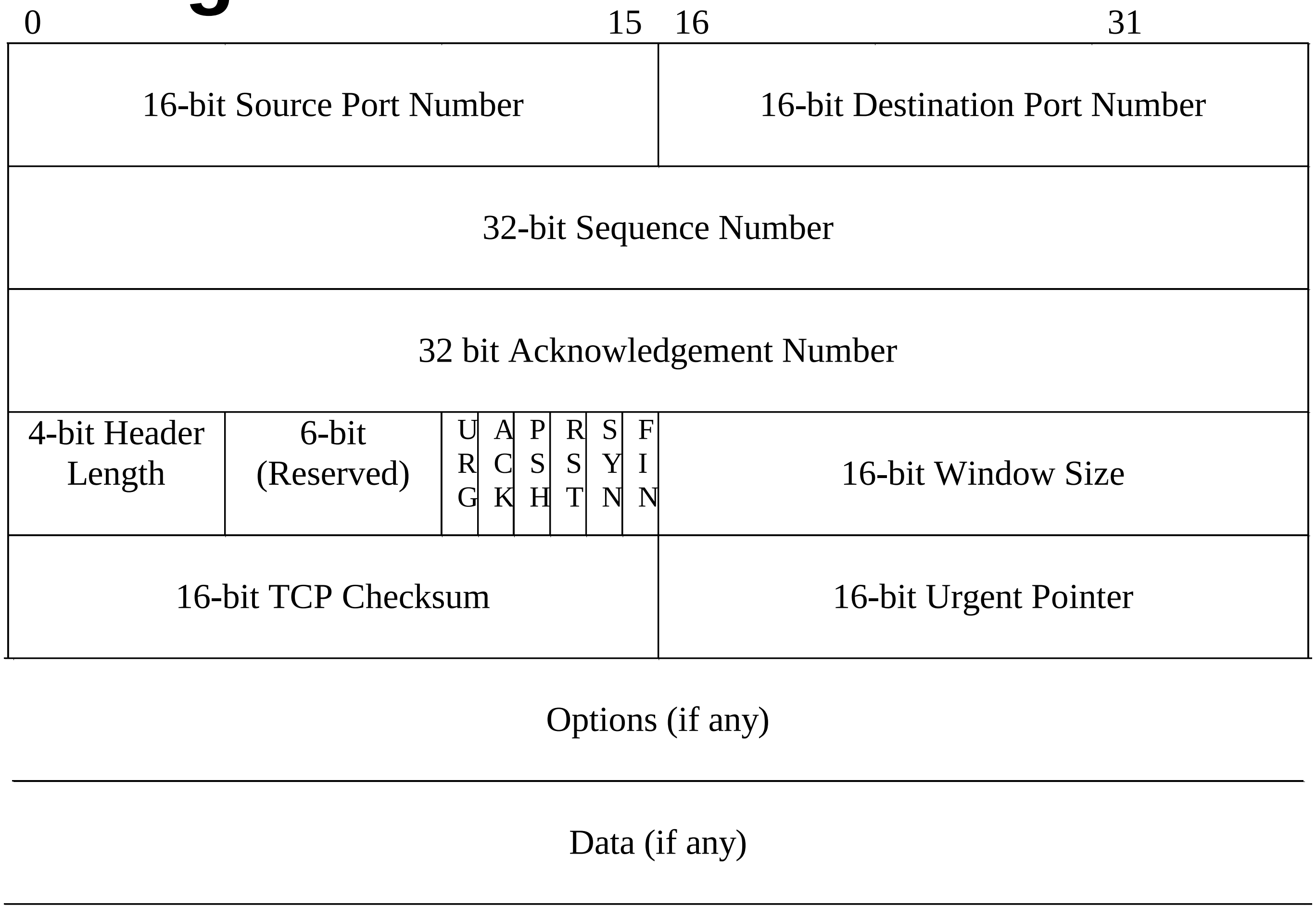
TCP (Transmission Control Protocol)

- **Connection-oriented, reliable protocol**
- Provides:
  1. **flow control** by providing sliding windows,
  2. **reliability** by providing sequence numbers and acknowledgments.
- TCP re-sends anything that is not received and supplies a **virtual circuit** between end-user applications.
- The advantage of TCP is that it provides **guaranteed delivery** of the segments.


Application
Header + data
UDP Header
Data
IP Header
Data
Frame Header
Frame Data
Frame Trailer


Application
Header + data
TCP Header
Data
IP Header
Data
Frame Header
Frame Data
Frame Trailer

TCP Segment Header



0 15 16 31

16-bit Source Port Number

16-bit Destination Port Number

32-bit Sequence Number

32 bit Acknowledgement Number

4-bit Header Length

6-bit (Reserved)

URG ACK PSH RST SYN FIN

16-bit Window Size

16-bit TCP Checksum

16-bit Urgent Pointer

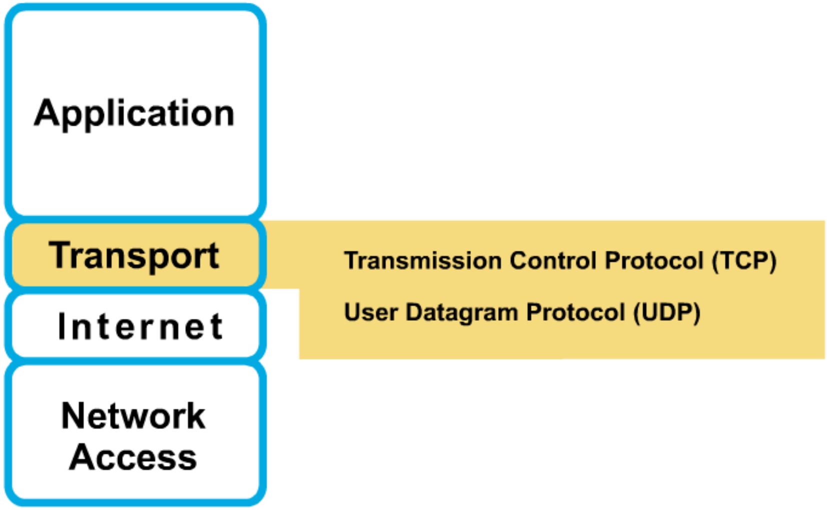
Options (if any)

Data (if any)

Some of the protocols that use TCP are:

- HTTP
- Telnet
- FTP

Transport Layer Overview



Application

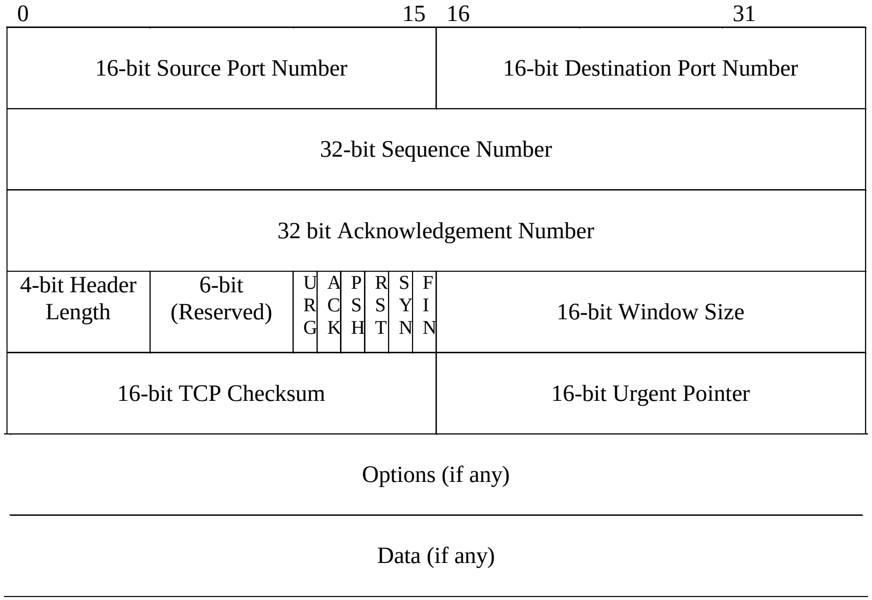
Transport

Internet

Network Access

Transmission Control Protocol (TCP)

User Datagram Protocol (UDP)



<table>
<tr><th colspan="3">0</th><th colspan="6">15 16</th><th>31</th></tr>
<tr><td colspan="3">16-bit Source Port Number</td><td colspan="7">16-bit Destination Port Number</td></tr>
<tr><td colspan="10">32-bit Sequence Number</td></tr>
<tr><td colspan="10">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="2">16-bit Window Size</td></tr>
<tr><td colspan="3">16-bit TCP Checksum</td><td colspan="7">16-bit Urgent Pointer</td></tr>
<tr><td colspan="10">Options (if any)</td></tr>
<tr><td colspan="10">Data (if any)</td></tr>
</table>

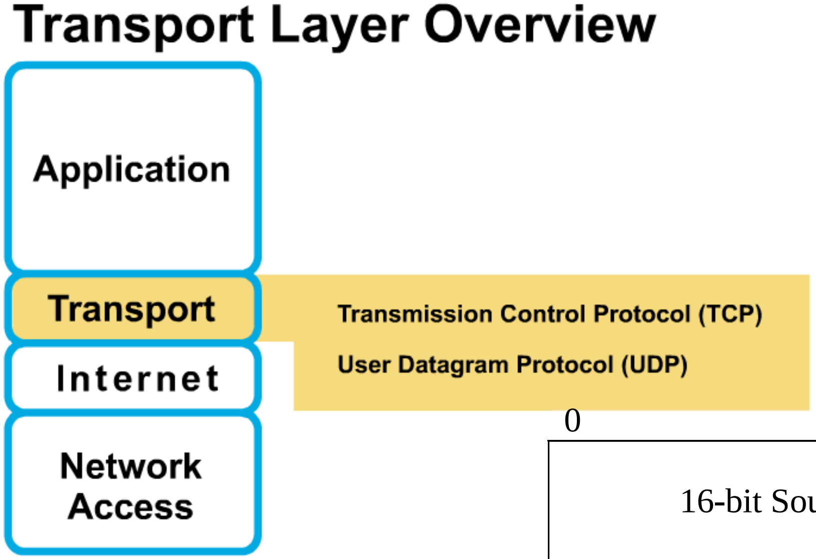
- **source port** -- the number of the calling port
- **destination port** -- the number of the called port
- **sequence number** -- the number used to ensure correct sequencing of the arriving data
- **acknowledgment number** -- the next expected TCP octet
- **HLEN** -- the number of 32-bit words in the header
- **reserved** -- set to 0
- **code bits** -- the control functions (e.g. setup and termination of a session)
- **window** -- the number of octets that the sender is willing to accept
- **checksum** -- the calculated checksum of the header and data fields
- **urgent pointer** -- indicates the end of the urgent data
- **option** -- one currently defined: maximum TCP segment size
- **data** -- upper-layer protocol data

UDP Segment Header

0 15	16 31
16-bit Source Port Number	16-bit Destination Port Number
16-bit UDP Length	16-bit UDP Checksum
Data (if any)	

- **UDP** -- ***connectionless and unreliable***; although responsible for transmitting messages, no software checking for segment delivery is provided at this layer.
- No flow control, no reliability.
- The advantage that UDP provides is **speed**.
- Since UDP provides no acknowledgments, less traffic is sent across the network, making the transfer faster.
- Protocols that use UDP include the following:
  - TFTP
  - SNMP
  - Network File System (NFS)
  - Domain Name System (DNS)

UDP Segment Header



0 15	16 31
16-bit Source Port Number	16-bit Destination Port Number
16-bit UDP Length	16-bit UDP Checksum
Data (if any)	

- **source port** -- the number of the calling port
- **destination port** -- the number of the called port
- **UDP length** -- the length of the UDP header
- **checksum** -- the calculated checksum of the header and data fields
- **data** -- upper-layer protocol data

Topics

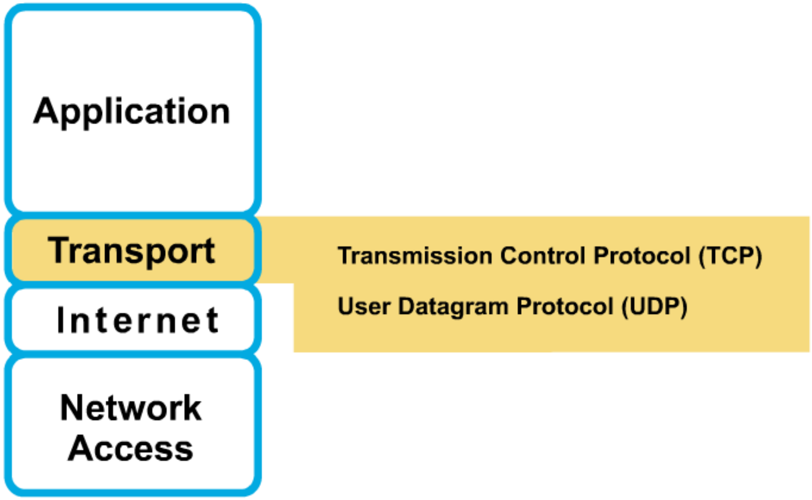
Layer 3 Concepts

- TCP/IP and the Internet Layer
- Diagram the IP datagram
- Internet Control Message Protocol (ICMP)

TCP/IP protocol stack and the transport layer

- TCP and UDP segment format
- TCP and UDP port numbers
- TCP three-way handshake/open connection
- TCP simple acknowledgment and windowing

Transport Layer Overview



Application

Transport

Internet

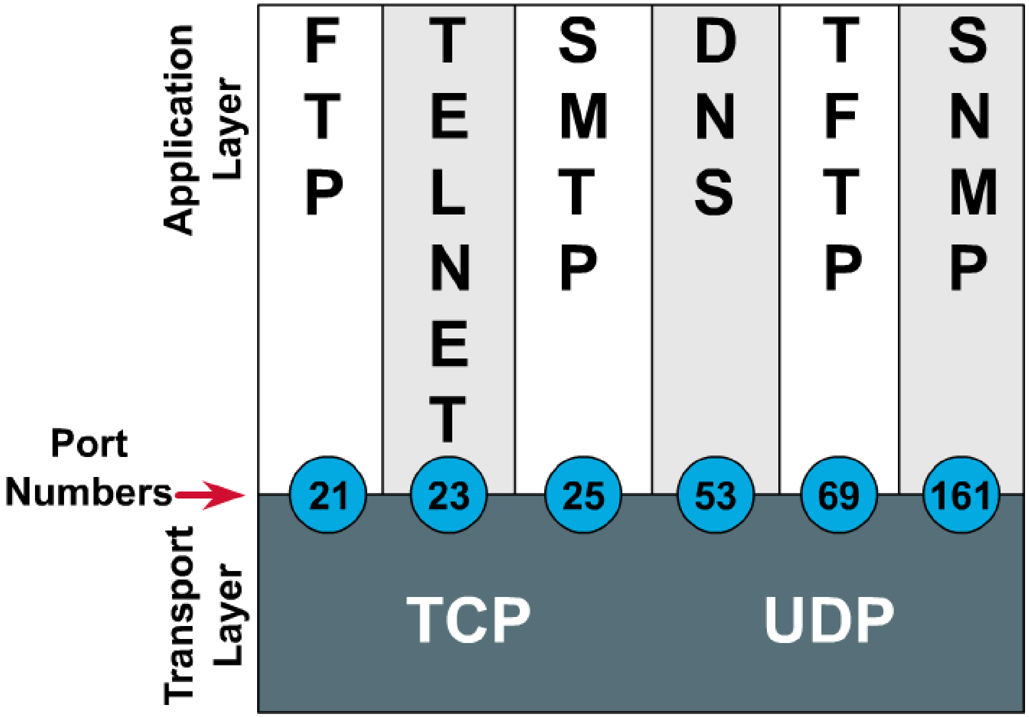
Network Access

Transmission Control Protocol (TCP)

User Datagram Protocol (UDP)

0								15	16	31
16-bit Source Port Number									16-bit Destination Port Number	
32-bit Sequence Number										
32 bit Acknowledgement Number										
4-bit Header Length	6-bit (Reserved)	U R G	A C K	P S H	R S T	S Y N	F I N		16-bit Window Size	
16-bit TCP Checksum									16-bit Urgent Pointer	
Options (if any)										
Data (if any)										

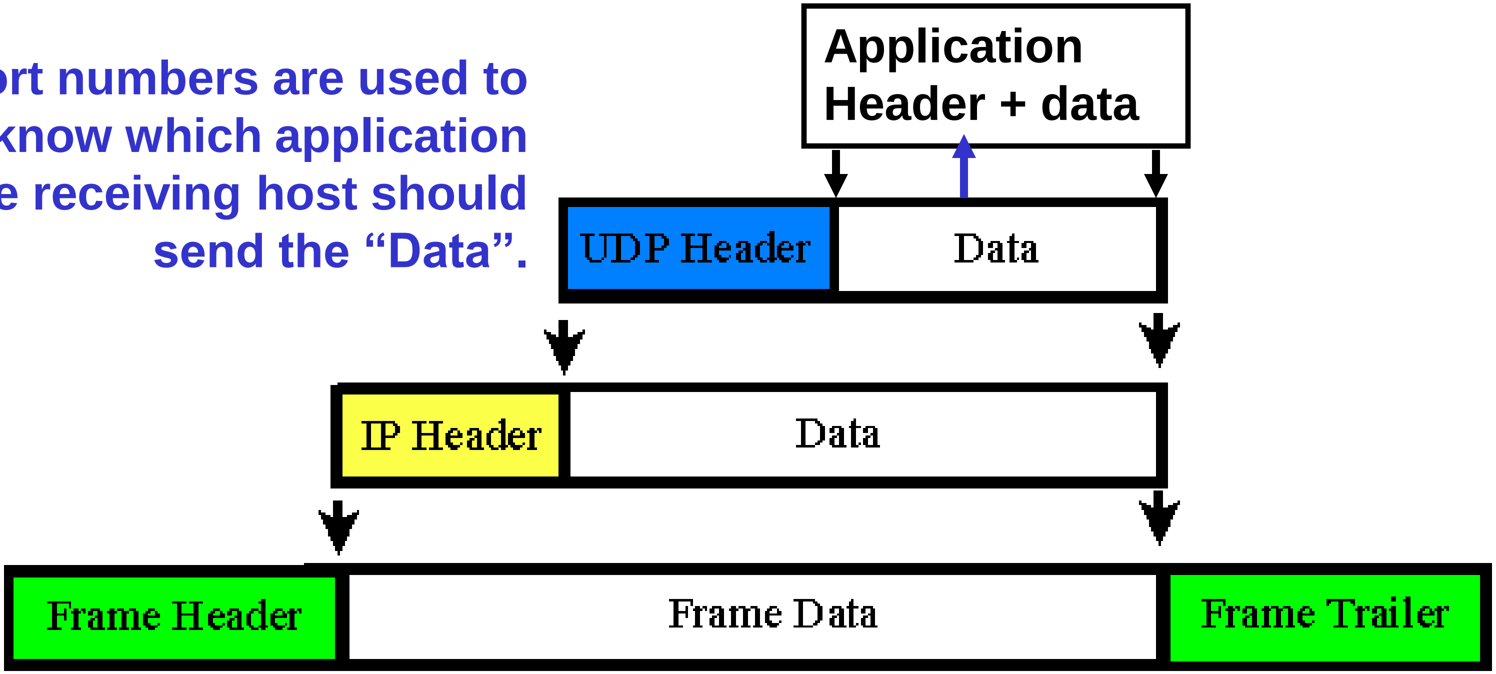
Port Numbers



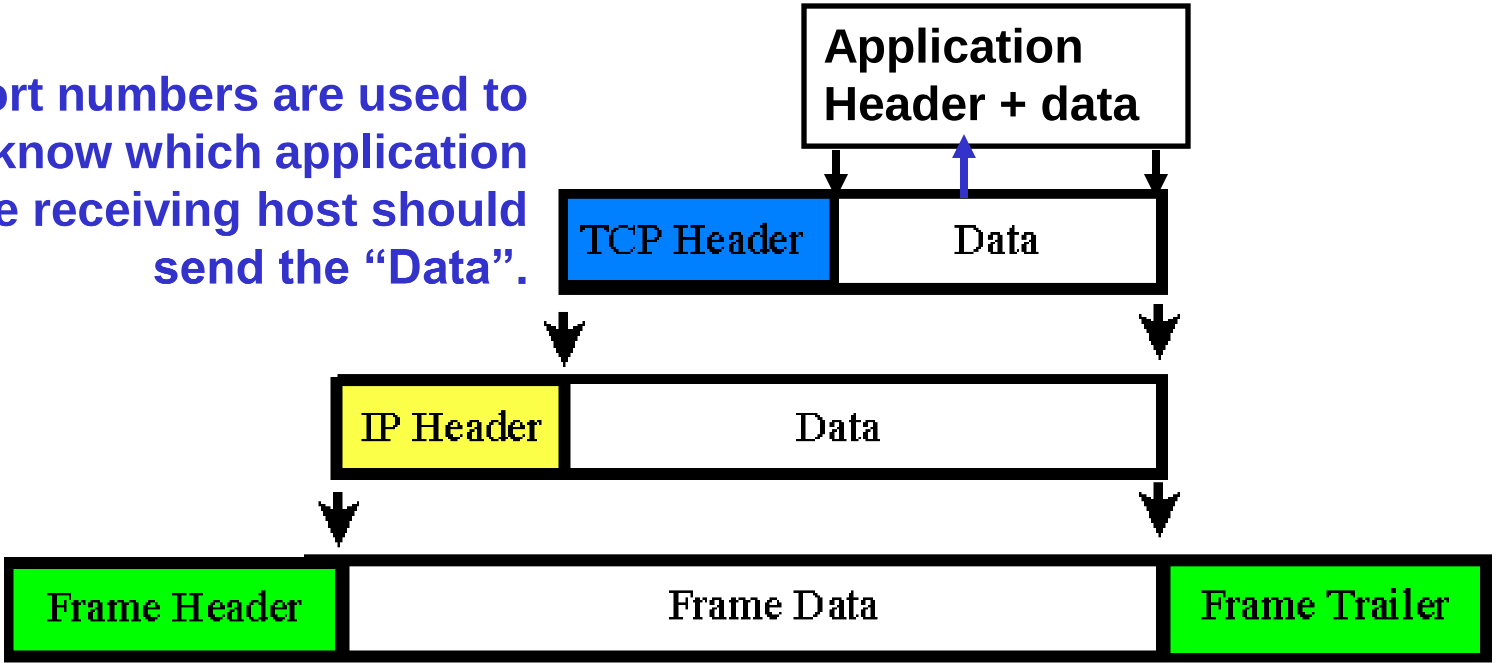
HTTP is Port 80

- **Both TCP and UDP** use ports (or sockets) numbers to pass information to the upper layers.

Port numbers are used to know which application the receiving host should send the “Data”.

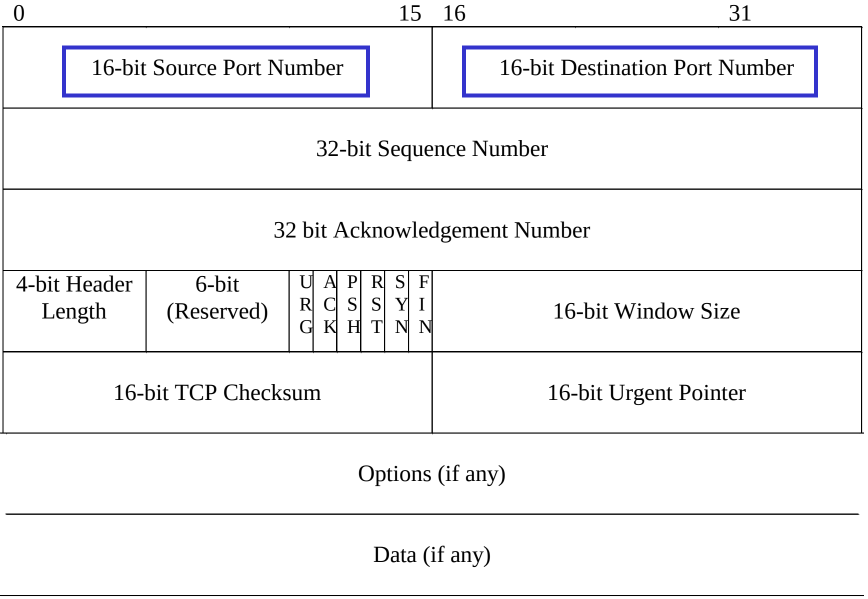


Port numbers are used to know which application the receiving host should send the “Data”.

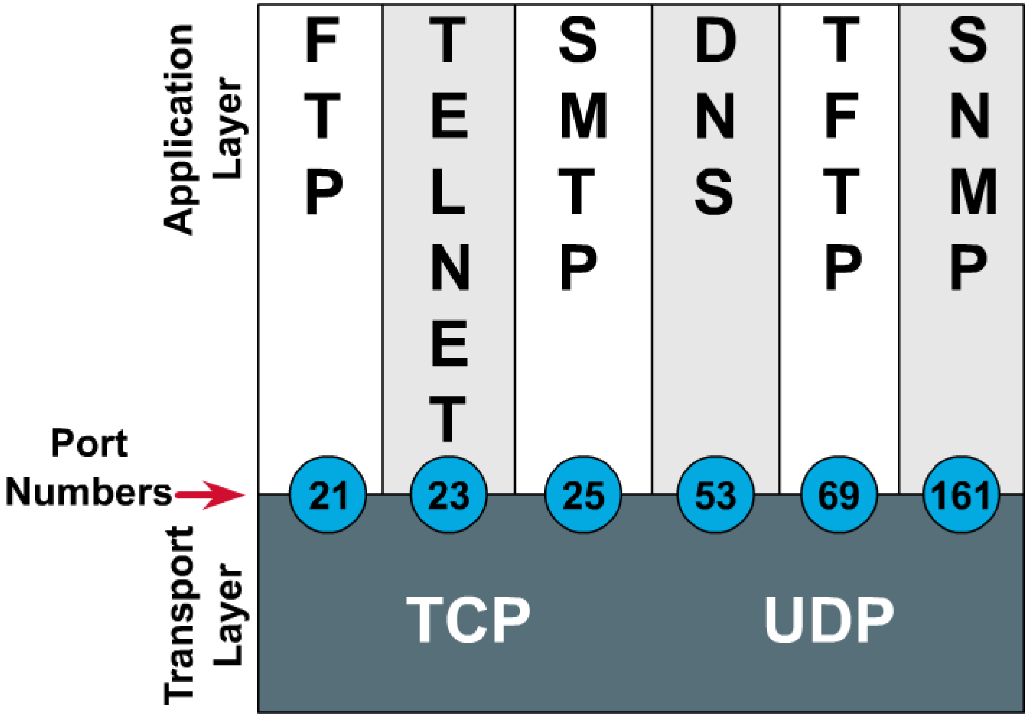


Port Numbers

TCP Header



0 15	16 31
16-bit Source Port Number	16-bit Destination Port Number
32-bit Sequence Number	
32 bit Acknowledgement Number	
4-bit Header Length 6-bit (Reserved) URG ACK PSH RST SYN FIN	16-bit Window Size
16-bit TCP Checksum	16-bit Urgent Pointer
Options (if any)	
Data (if any)	



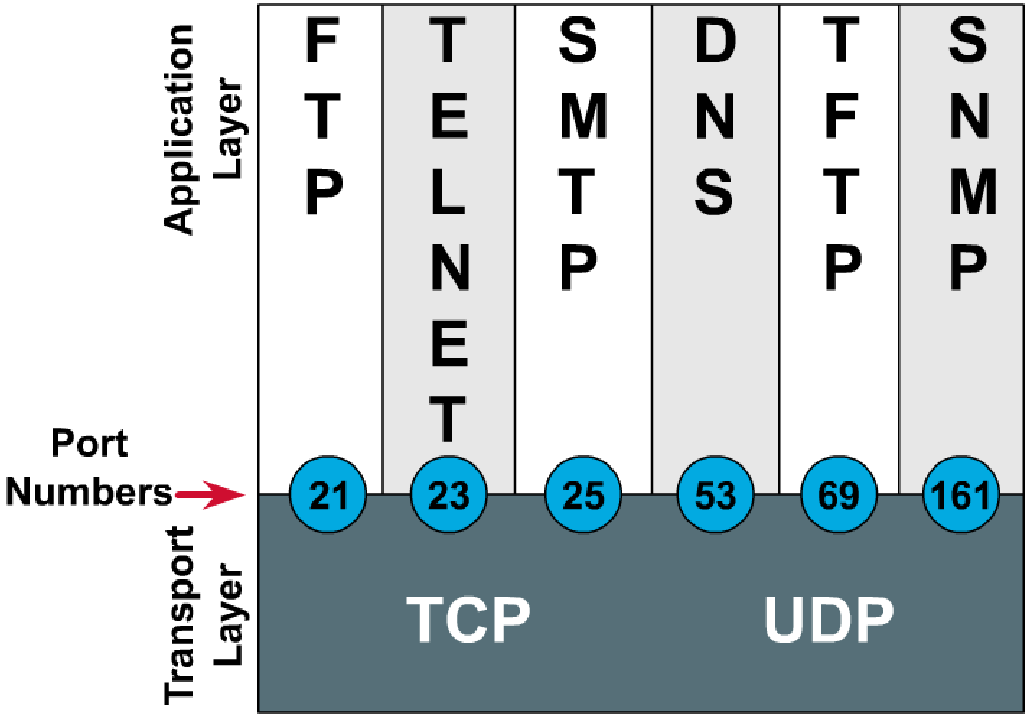
HTTP is Port 80

- Application software developers have agreed to use the **well-known port numbers** that are defined in RFC 1700.
- For example, any conversation bound for an **FTP** application uses the standard port number **21**.

Port Numbers

TCP Header

<table>
<tr><td colspan="8">0 15</td><td>16 31</td></tr>
<tr><td colspan="8">16-bit Source Port Number</td><td>16-bit Destination Port Number</td></tr>
<tr><td colspan="9">32-bit Sequence Number</td></tr>
<tr><td colspan="9">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16-bit Window Size</td></tr>
<tr><td colspan="8">16-bit TCP Checksum</td><td>16-bit Urgent Pointer</td></tr>
<tr><td colspan="9">Options (if any)</td></tr>
<tr><td colspan="9">Data (if any)</td></tr>
</table>



HTTP is Port 80

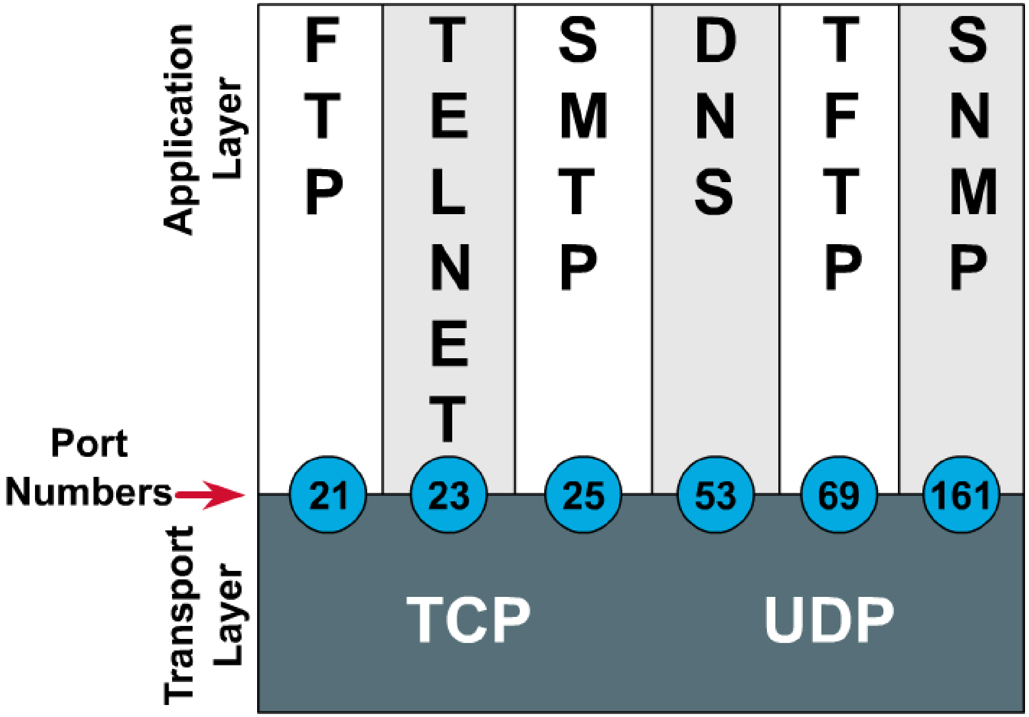
- Conversations that do not involve an application with a well-known port number are, instead, assigned port numbers that are randomly selected from within a specific range.
- These port numbers are used as source and destination addresses in the TCP segment.

Port Numbers

TCP Header

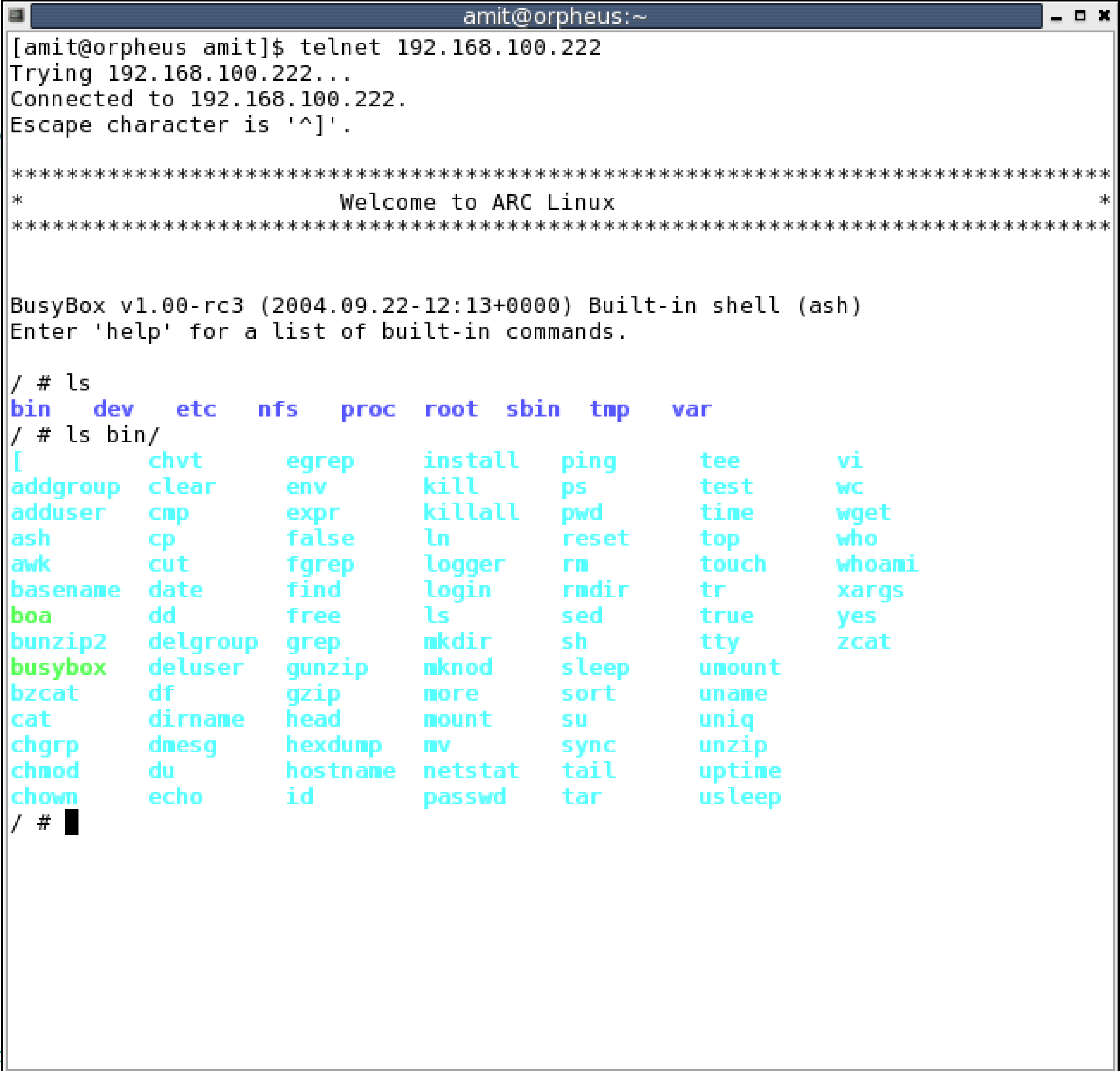
<table>
<tr><td colspan="8">0 15</td><td>16 31</td></tr>
<tr><td colspan="8">16-bit Source Port Number</td><td>16-bit Destination Port Number</td></tr>
<tr><td colspan="9">32-bit Sequence Number</td></tr>
<tr><td colspan="9">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16-bit Window Size</td></tr>
<tr><td colspan="8">16-bit TCP Checksum</td><td>16-bit Urgent Pointer</td></tr>
<tr><td colspan="9">Options (if any)</td></tr>
<tr><td colspan="9">Data (if any)</td></tr>
</table>

HTTP is Port 80



- Some ports are reserved in both TCP and UDP, although applications might not be written to support them.
- The range for assigned ports managed by the IANA is 0-1023.: http://www.iana.org/assignments/port-numbers
  - The **Well Known Ports** are those from **0 through 1023.** (This is updated information as of 11-13-2002. Before then, 0 – 255 were considered well known ports.)
  - The **Registered Ports** are those from **1024 through 49151**
  - The **Dynamic and/or Private Ports** are those from **49152 through 65535**

Telnet



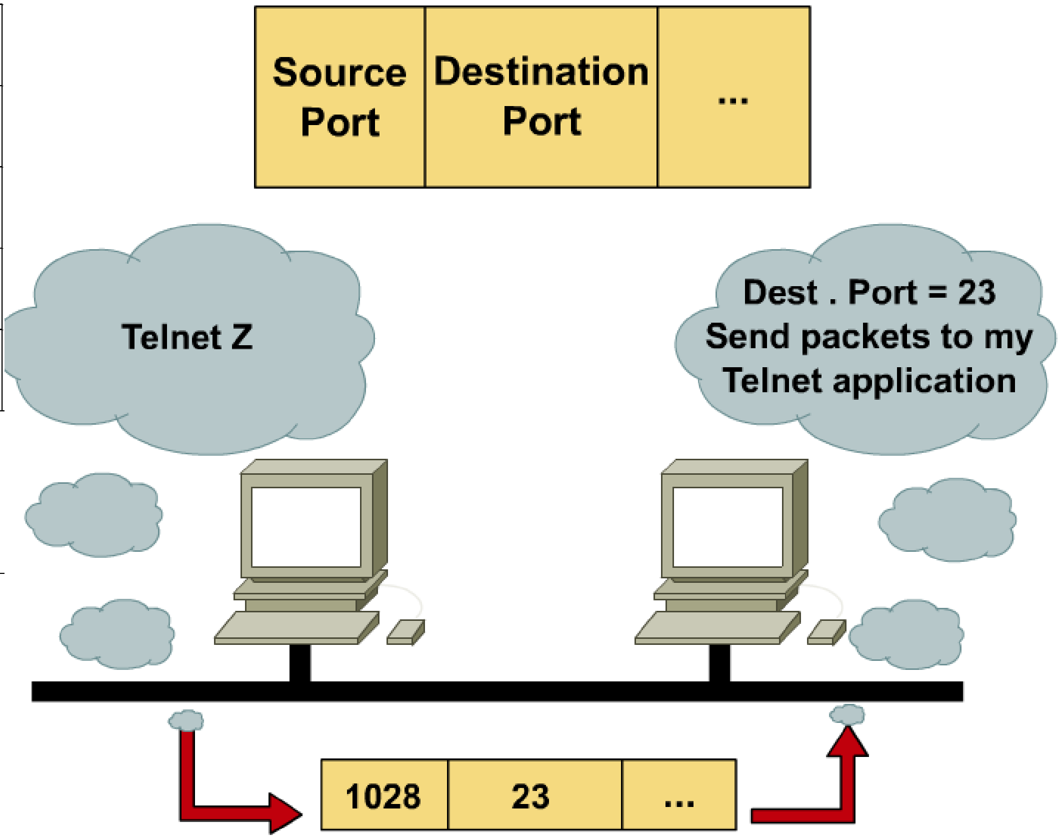
```
[amit@orpheus amit]$ telnet 192.168.100.222
Trying 192.168.100.222...
Connected to 192.168.100.222.
Escape character is '^]'.

********************************************************************************
*                       Welcome to ARC Linux                                   *
********************************************************************************



BusyBox v1.00-rc3 (2004.09.22-12:13+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ # ls
bin    dev    etc    nfs    proc   root   sbin   tmp    var
/ # ls bin/
[          chvt       egrep      install    ping       tee        vi
addgroup   clear      env        kill       ps         test       wc
adduser    cmp        expr       killall    pwd        time       wget
ash        cp         false      ln         reset      top        who
awk        cut        fgrep      logger     rm         touch      whoami
basename   date       find       login      rmdir      tr         xargs
boa        dd         free       ls         sed        true       yes
bunzip2    delgroup   grep       mkdir      sh         tty        zcat
busybox    deluser    gunzip     mknod      sleep      umount
bzcat      df         gzip       more       sort       uname
cat        dirname    head       mount      su         uniq
chgrp      dmesg      hexdump    mv         sync       unzip
chmod      du         hostname   netstat    tail       uptime
chown      echo       id         passwd     tar        usleep
/ # 
```

TCP Header

31

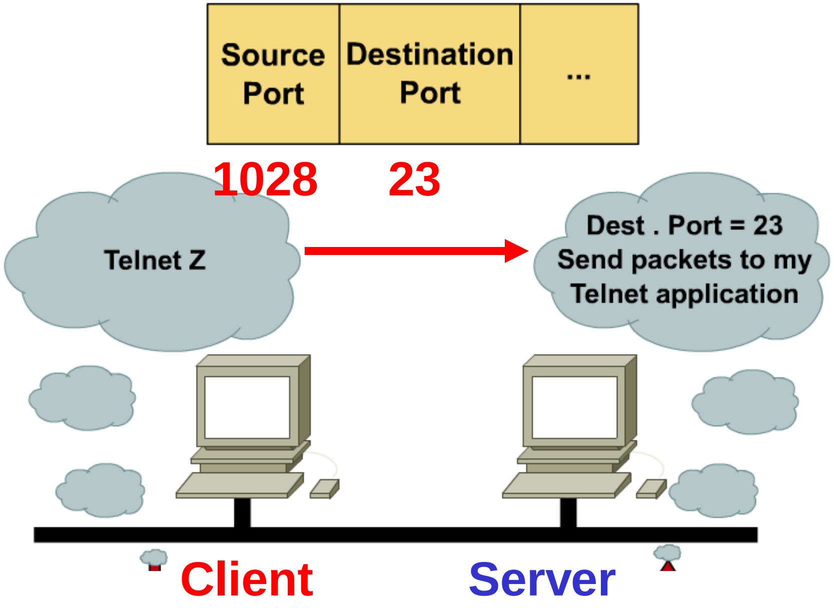
16-bit Source Port Number								16-bit Destination Port Number
32-bit Sequence Number								
32 bit Acknowledgement Number								
4-bit Header Length	6-bit (Reserved)	URG	ACK	PSH	RST	SYN	FIN	16-bit Window Size
16-bit TCP Checksum								16-bit Urgent Pointer
Options (if any)								
Data (if any)								

TCP/UDP Port Numbers

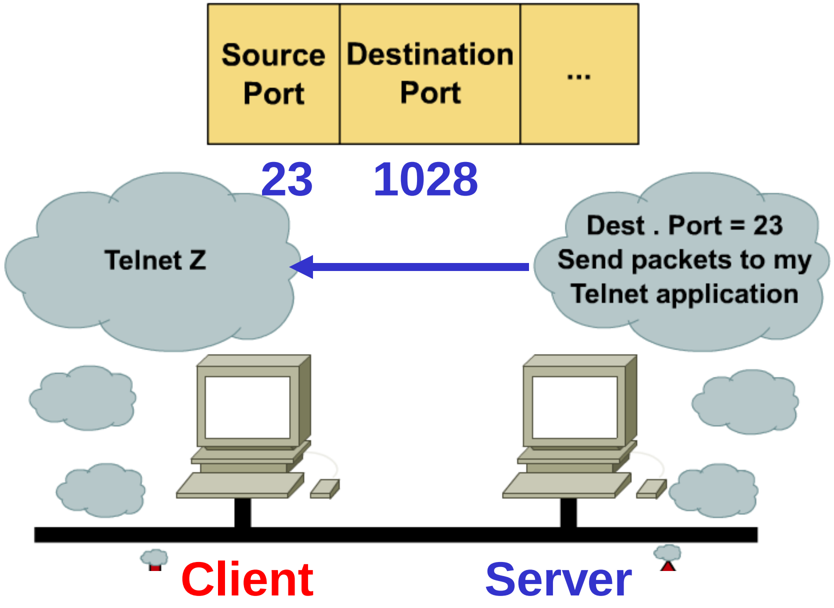


- End systems use port numbers to select the proper application.
- Originating source port numbers, usually some numbers larger than 1023, are dynamically assigned by the source host.

TCP/UDP Port Numbers



TCP/UDP Port Numbers



Notice the difference in how source and destination port numbers are used with clients and servers:

Client (initiating Telnet service):

- **Destination Port = 23 (telnet)**
- **Source Port = 1028 (dynamically assigned)**

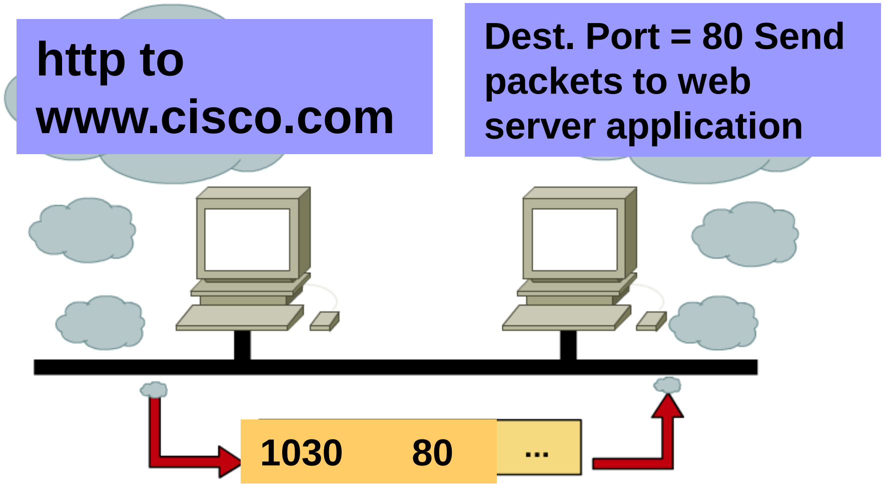
Server (responding to Telnet service):

- **Destination Port = 1028 (source port of client)**
- **Source Port = 23 (telnet)**

TCP/UDP Port Numbers

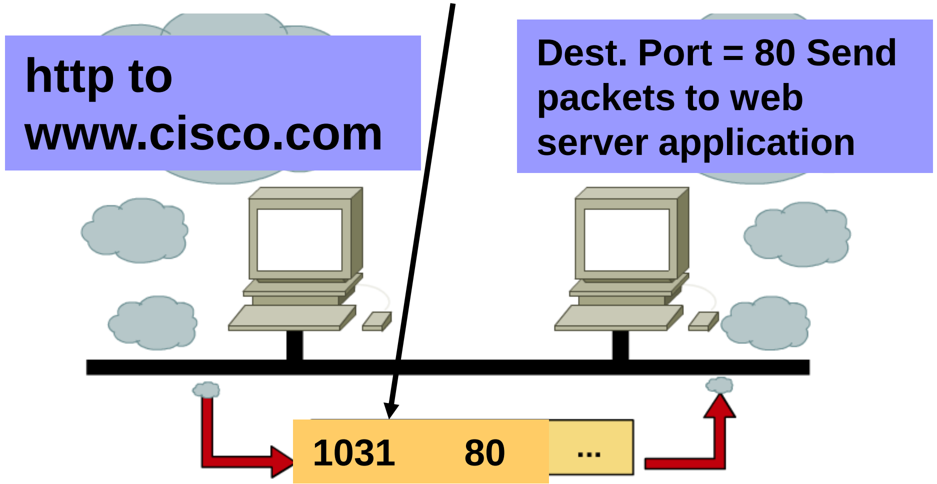
Source Port	Destination Port	...

Second http session from the between the same client and server. Same destination port, but different source port to uniquely identify this web session.



http to www.cisco.com

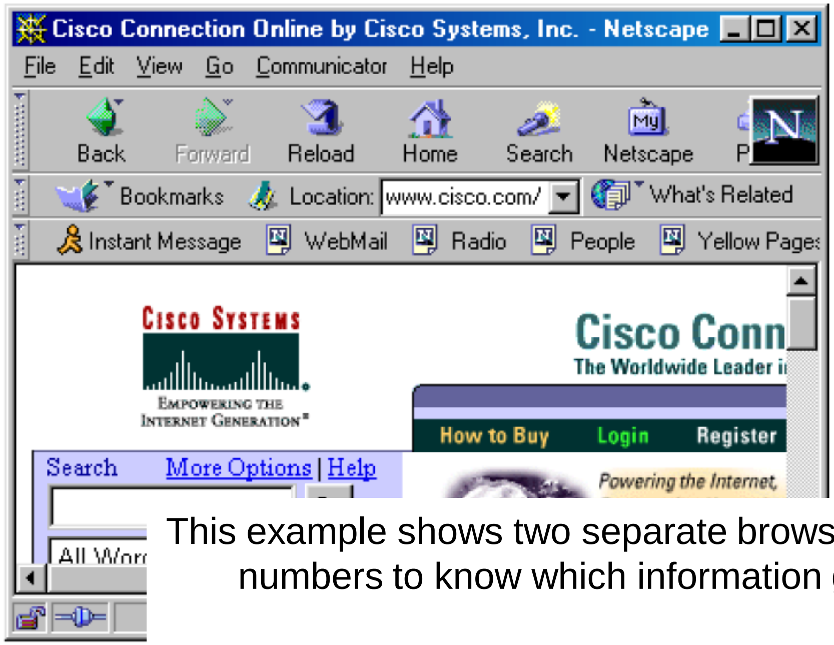
Dest. Port = 80 Send packets to web server application



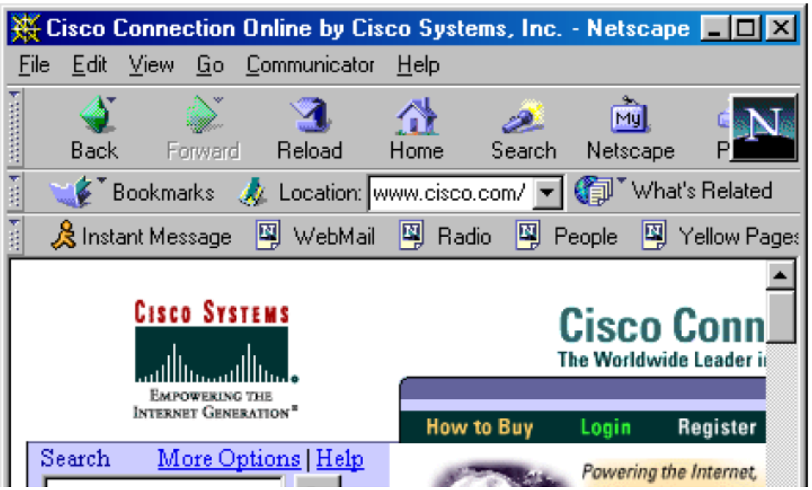
http to www.cisco.com

Dest. Port = 80 Send packets to web server application

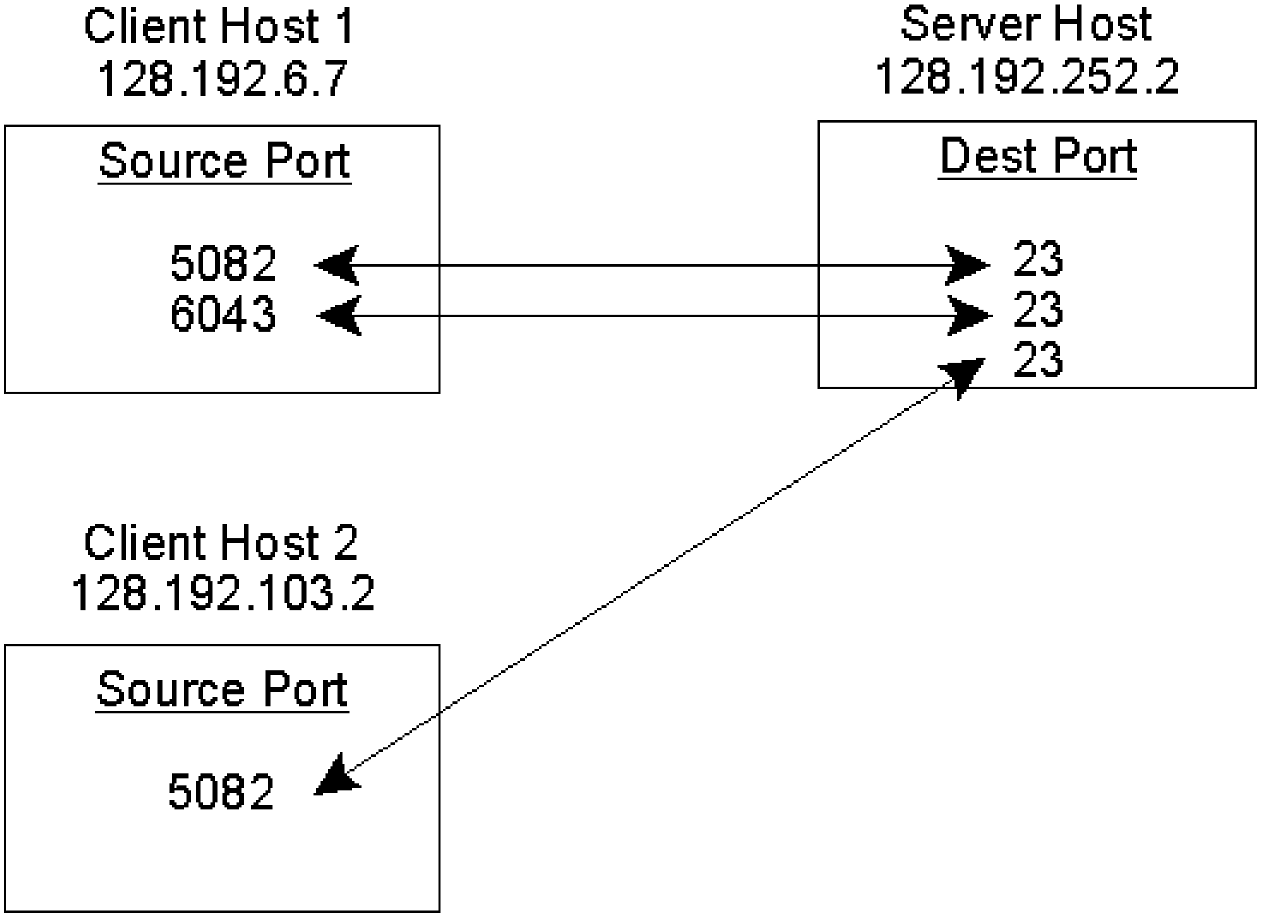
Netscape Navigator



Netscape Navigator

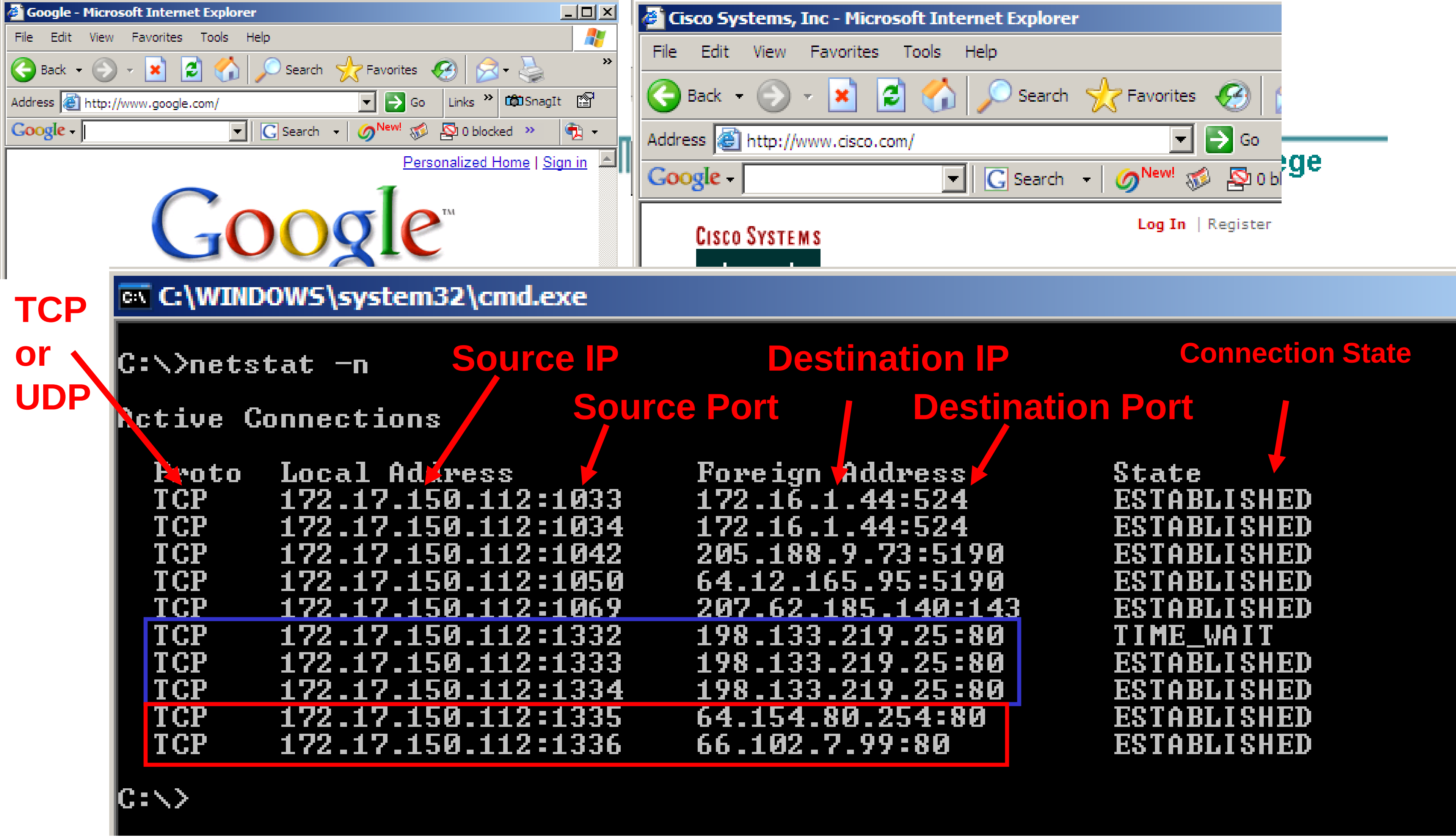


This example shows two separate browser windows to the same URL. TCP/IP uses source port numbers to know which information goes to which window.



What makes each connection unique?

- Connection defined by the pair of numbers:
  - **Source IP address**, **Source port**
  - **Destination IP address**, **Destination port**
- Different connections can use the same destination port on server host as long as the source ports or source IPs are different.


Google - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.google.com/ Go Links SnagIt
Google Search New! 0 blocked
Personalized Home | Sign in
Google
Cisco Systems, Inc - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.cisco.com/ Go
Google Search New! 0 bl
Log In | Register
Cisco Systems
C:\WINDOWS\system32\cmd.exe
TCP or UDP
Source IP
Destination IP
Connection State
Source Port
Destination Port

```
C:\>netstat -n

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    172.17.150.112:1033    172.16.1.44:524        ESTABLISHED
  TCP    172.17.150.112:1034    172.16.1.44:524        ESTABLISHED
  TCP    172.17.150.112:1042    205.188.9.73:5190      ESTABLISHED
  TCP    172.17.150.112:1050    64.12.165.95:5190      ESTABLISHED
  TCP    172.17.150.112:1069    207.62.185.140:143     ESTABLISHED
  TCP    172.17.150.112:1332    198.133.219.25:80      TIME_WAIT
  TCP    172.17.150.112:1333    198.133.219.25:80      ESTABLISHED
  TCP    172.17.150.112:1334    198.133.219.25:80      ESTABLISHED
  TCP    172.17.150.112:1335    64.154.80.254:80       ESTABLISHED
  TCP    172.17.150.112:1336    66.102.7.99:80         ESTABLISHED

C:\>
```

www.google.com **www.cisco.com** **netstat –n**

- **Note:** In actuality, when you open up a single html page, there are usually several TCP sessions created, not just one.
- Example of multiple TCP connections for a single http session.

Topics

Layer 3 Concepts

- TCP/IP and the Internet Layer
- Diagram the IP datagram
- Internet Control Message Protocol (ICMP)

TCP/IP protocol stack and the transport layer

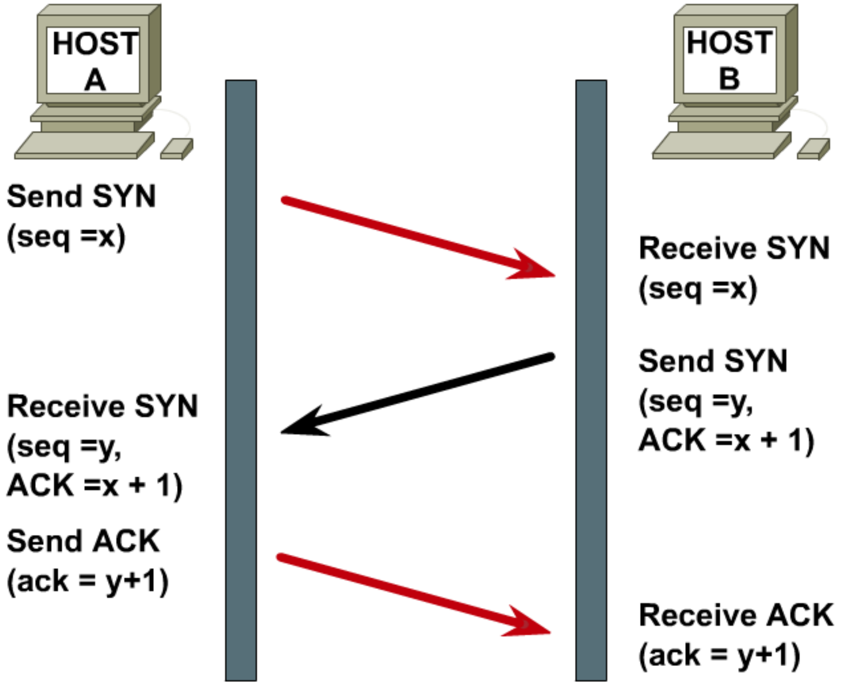
- TCP and UDP segment format
- TCP and UDP port numbers
- TCP three-way handshake/open connection
- TCP simple acknowledgment and windowing

TCP Header

31

<table>
<tr><td colspan="8">16-bit Source Port Number</td><td>16-bit Destination Port Number</td></tr>
<tr><td colspan="9">32-bit Sequence Number</td></tr>
<tr><td colspan="9">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16-bit Window Size</td></tr>
<tr><td colspan="8">16-bit TCP Checksum</td><td>16-bit Urgent Pointer</td></tr>
<tr><td colspan="9">Options (if any)</td></tr>
<tr><td colspan="9">Data (if any)</td></tr>
</table>

TCP Three-Way Handshake/ Open Connection

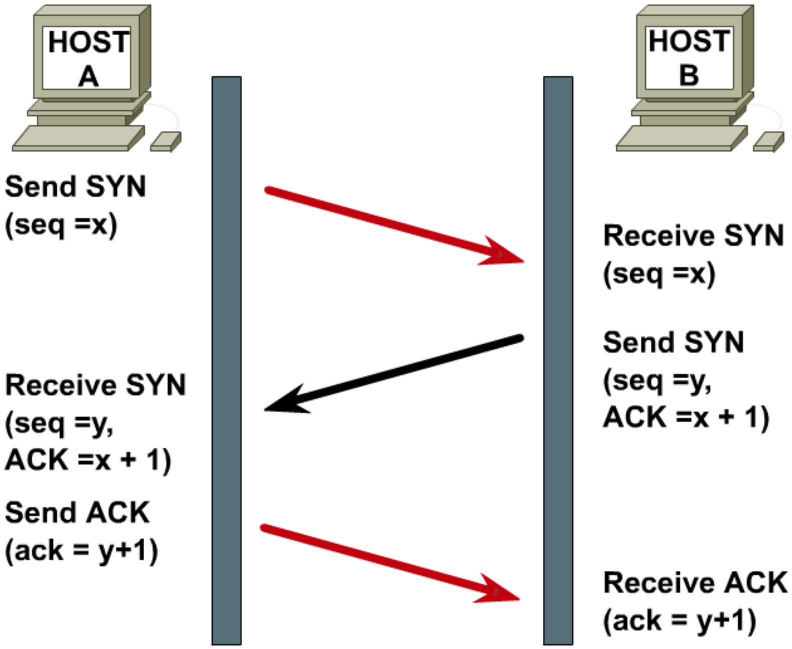


- For a connection to be established, the two end stations must synchronize on each other's TCP initial sequence numbers (ISNs).
- Sequence numbers are used to track the order of packets and to ensure that no packets are lost in transmission.
- The initial sequence number is the starting number used when a TCP connection is established.
- Exchanging beginning sequence numbers during the connection sequence ensures that lost data can be recovered.

TCP Header

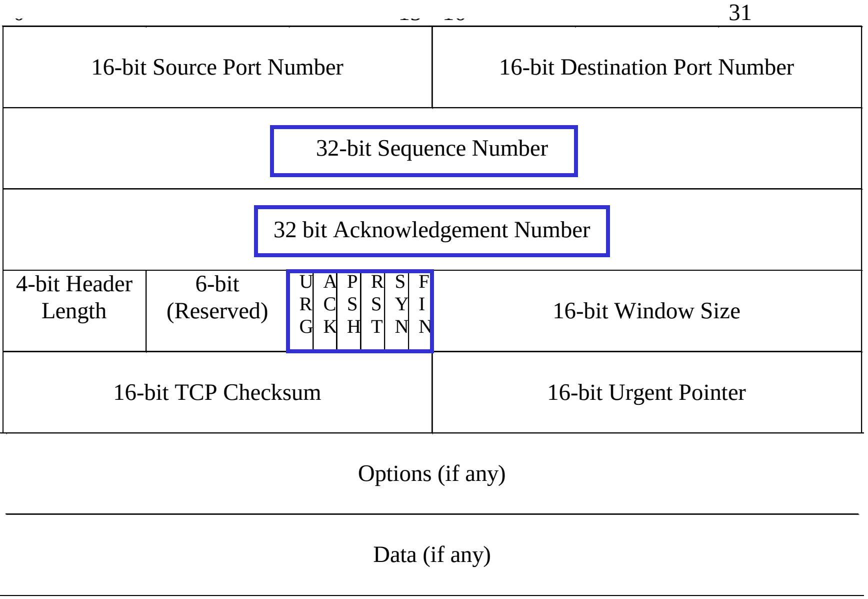
<table>
<tr><td colspan="16">31</td></tr>
<tr><td colspan="8">16-bit Source Port Number</td><td colspan="8">16-bit Destination Port Number</td></tr>
<tr><td colspan="16">32-bit Sequence Number</td></tr>
<tr><td colspan="16">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td colspan="8">16-bit Window Size</td></tr>
<tr><td colspan="8">16-bit TCP Checksum</td><td colspan="8">16-bit Urgent Pointer</td></tr>
<tr><td colspan="16">Options (if any)</td></tr>
<tr><td colspan="16">Data (if any)</td></tr>
</table>

TCP Three-Way Handshake/ Open Connection



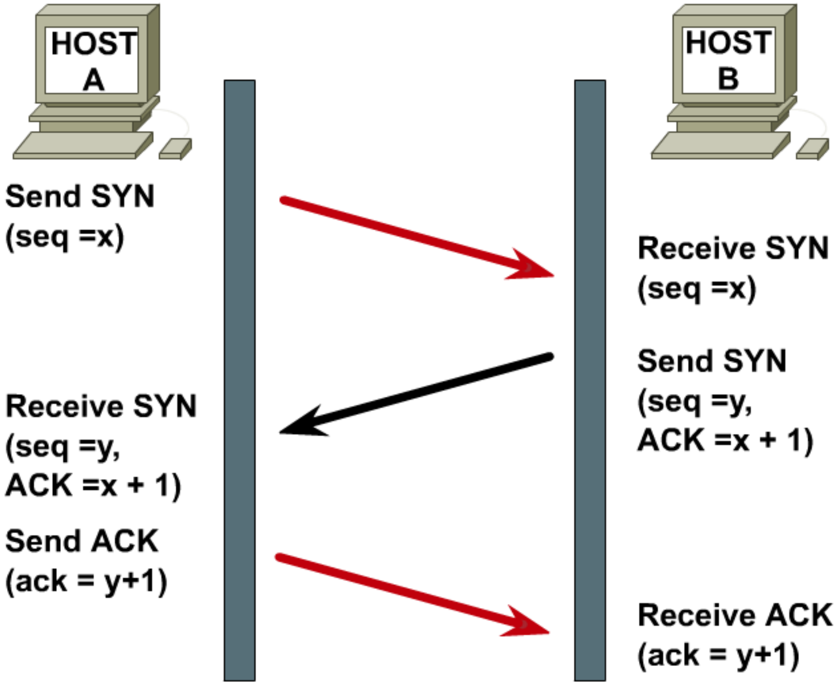
- ***Synchronization*** is accomplished by exchanging segments carrying the ISNs and a control bit called *SYN*, which stands for *synchronize*. (Segments carrying the SYN bit are also called SYNs.)
- Successful connection requires a suitable mechanism for choosing an initial sequence and a slightly involved handshake to exchange the ISNs.
- Synchronization requires that each side send its own ISN and receive a confirmation and ISN from the other side of the connection.
- Each side must receive the other side's ISN and send a confirming acknowledgment (ACK) in a specific order.

TCP Header



31				
16-bit Source Port Number			16-bit Destination Port Number	
32-bit Sequence Number				
32 bit Acknowledgement Number				
4-bit Header Length	6-bit (Reserved)	URG ACK PSH RST SYN FIN	16-bit Window Size	
16-bit TCP Checksum			16-bit Urgent Pointer	
Options (if any)				
Data (if any)				

TCP Three-Way Handshake/ Open Connection

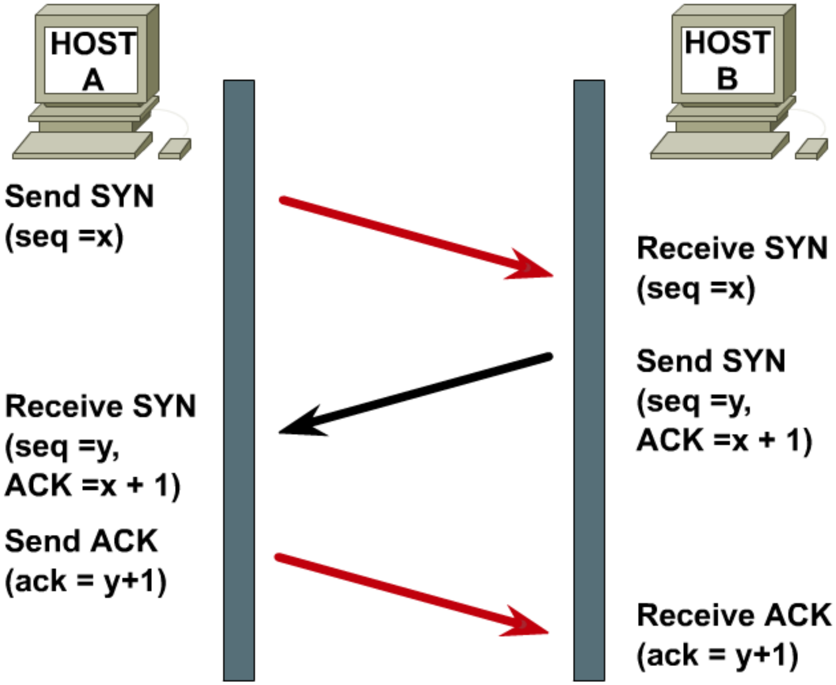


- Because the second and third steps can be combined in a single message, the exchange is called a three-way handshake/open connection.
- A three-way handshake is necessary because TCPs may use different mechanisms for picking the ISN.
- The receiver of the first SYN has no way of knowing if the segment was an old delayed one unless it remembers the last sequence number used on the connection, which is not always possible, and so it must ask the sender to verify this SYN

TCP Header

<table>
<tr><td colspan="4" align="right">31</td></tr>
<tr><td colspan="3">16-bit Source Port Number</td><td>16-bit Destination Port Number</td></tr>
<tr><td colspan="4">32-bit Sequence Number</td></tr>
<tr><td colspan="4">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>URG | ACK | PSH | RST | SYN | FIN</td><td>16-bit Window Size</td></tr>
<tr><td colspan="3">16-bit TCP Checksum</td><td>16-bit Urgent Pointer</td></tr>
<tr><td colspan="4">Options (if any)</td></tr>
<tr><td colspan="4">Data (if any)</td></tr>
</table>

TCP Three-Way Handshake/ Open Connection



Send SYN (seq =x)

Receive SYN (seq =x)

Send SYN (seq =y, ACK =x + 1)

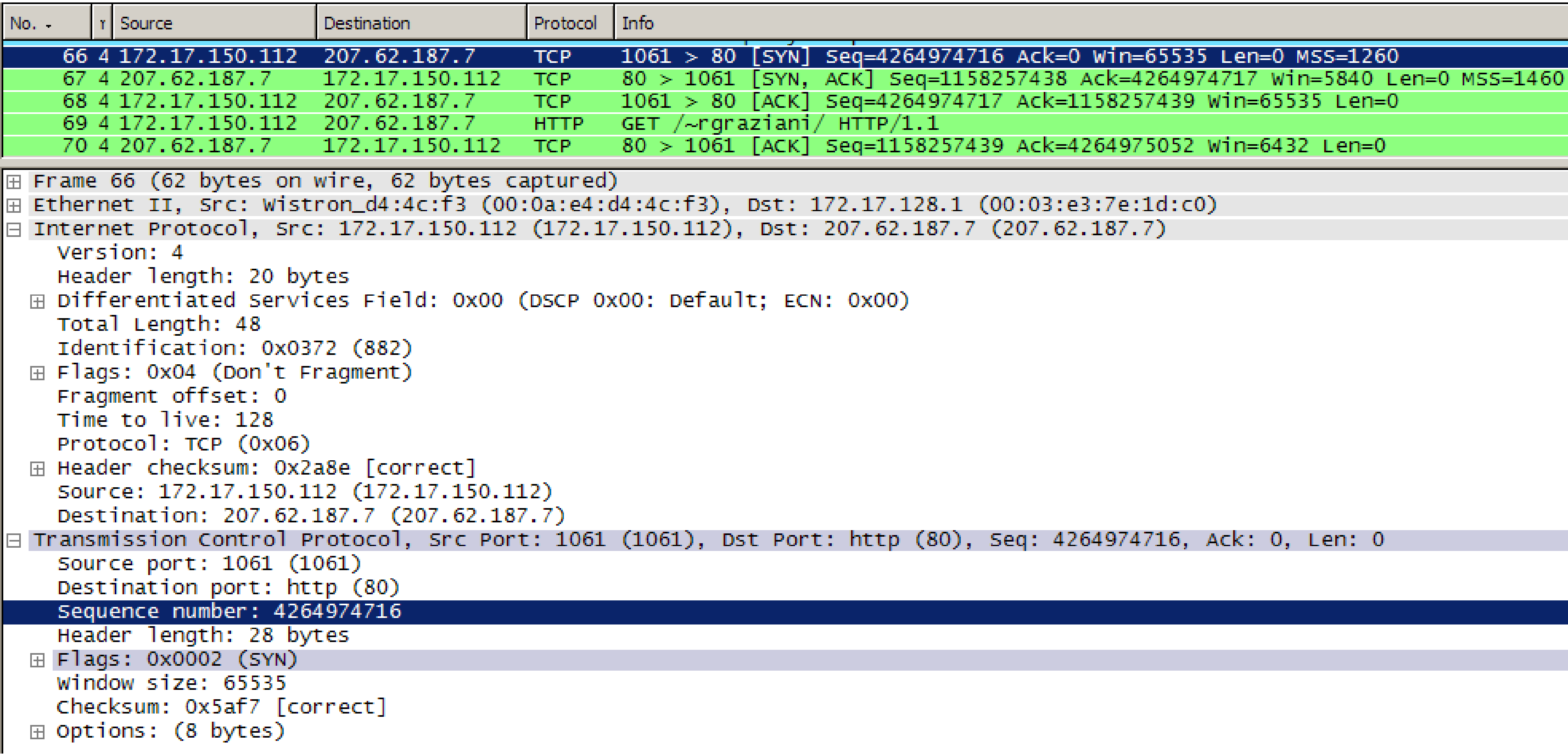
Receive SYN (seq =y, ACK =x + 1)

Send ACK (ack = y+1)

Receive ACK (ack = y+1)

- At this point, either side can begin communicating, and either side can break the communication because TCP is a peer-to-peer (balanced) communication method.

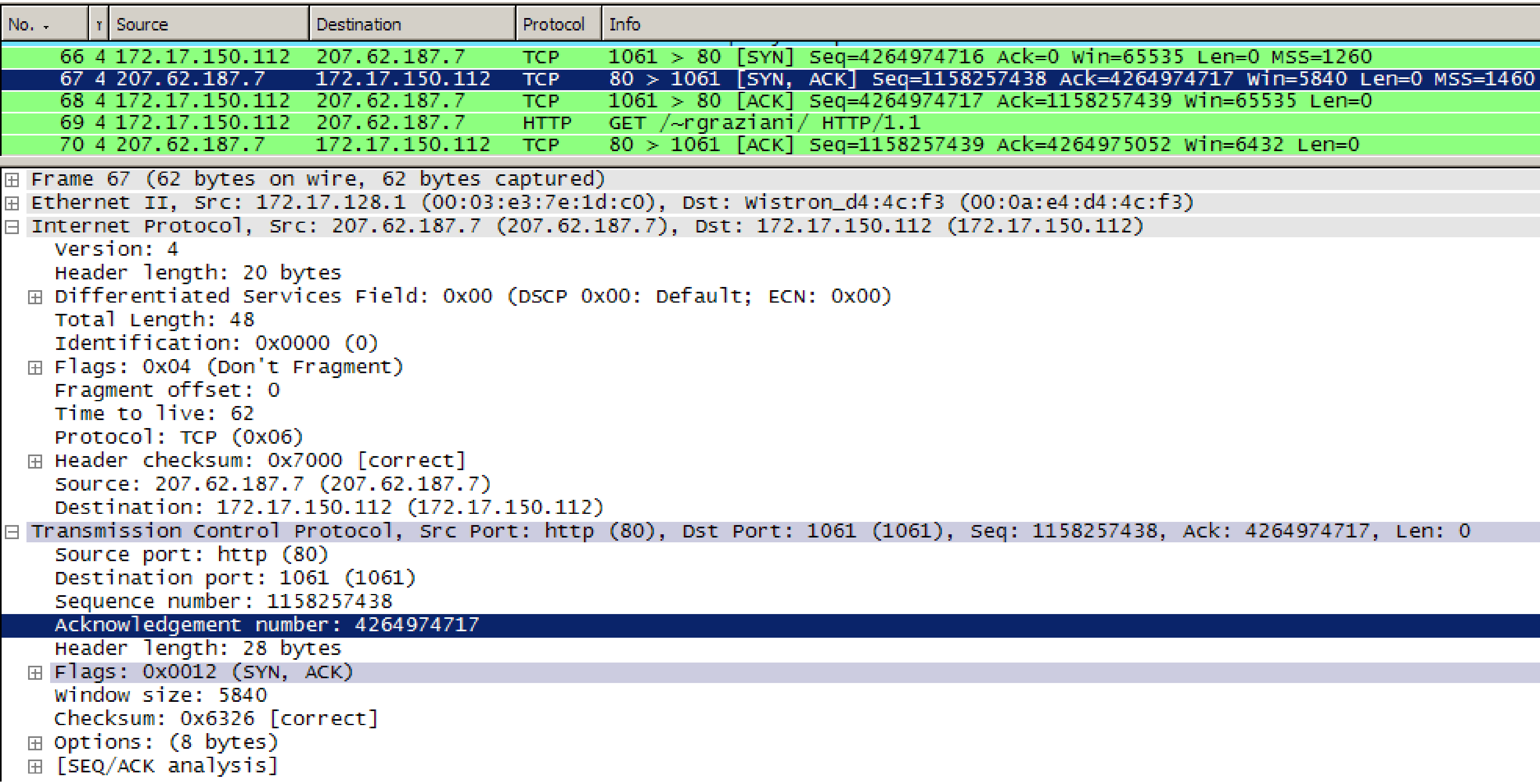
Client: Seq = 4264974716



```
No. -  T Source          Destination     Protocol  Info
    66 4 172.17.150.112  207.62.187.7    TCP       1061 > 80 [SYN] Seq=4264974716 Ack=0 Win=65535 Len=0 MSS=1260
    67 4 207.62.187.7    172.17.150.112  TCP       80 > 1061 [SYN, ACK] Seq=1158257438 Ack=4264974717 Win=5840 Len=0 MSS=1460
    68 4 172.17.150.112  207.62.187.7    TCP       1061 > 80 [ACK] Seq=4264974717 Ack=1158257439 Win=65535 Len=0
    69 4 172.17.150.112  207.62.187.7    HTTP      GET /~rgraziani/ HTTP/1.1
    70 4 207.62.187.7    172.17.150.112  TCP       80 > 1061 [ACK] Seq=1158257439 Ack=4264975052 Win=6432 Len=0
```

```
⊞ Frame 66 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: Wistron_d4:4c:f3 (00:0a:e4:d4:4c:f3), Dst: 172.17.128.1 (00:03:e3:7e:1d:c0)
⊟ Internet Protocol, Src: 172.17.150.112 (172.17.150.112), Dst: 207.62.187.7 (207.62.187.7)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 48
    Identification: 0x0372 (882)
  ⊞ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
  ⊞ Header checksum: 0x2a8e [correct]
    Source: 172.17.150.112 (172.17.150.112)
    Destination: 207.62.187.7 (207.62.187.7)
⊟ Transmission Control Protocol, Src Port: 1061 (1061), Dst Port: http (80), Seq: 4264974716, Ack: 0, Len: 0
    Source port: 1061 (1061)
    Destination port: http (80)
    Sequence number: 4264974716
    Header length: 28 bytes
  ⊞ Flags: 0x0002 (SYN)
    Window size: 65535
    Checksum: 0x5af7 [correct]
  ⊞ Options: (8 bytes)
```

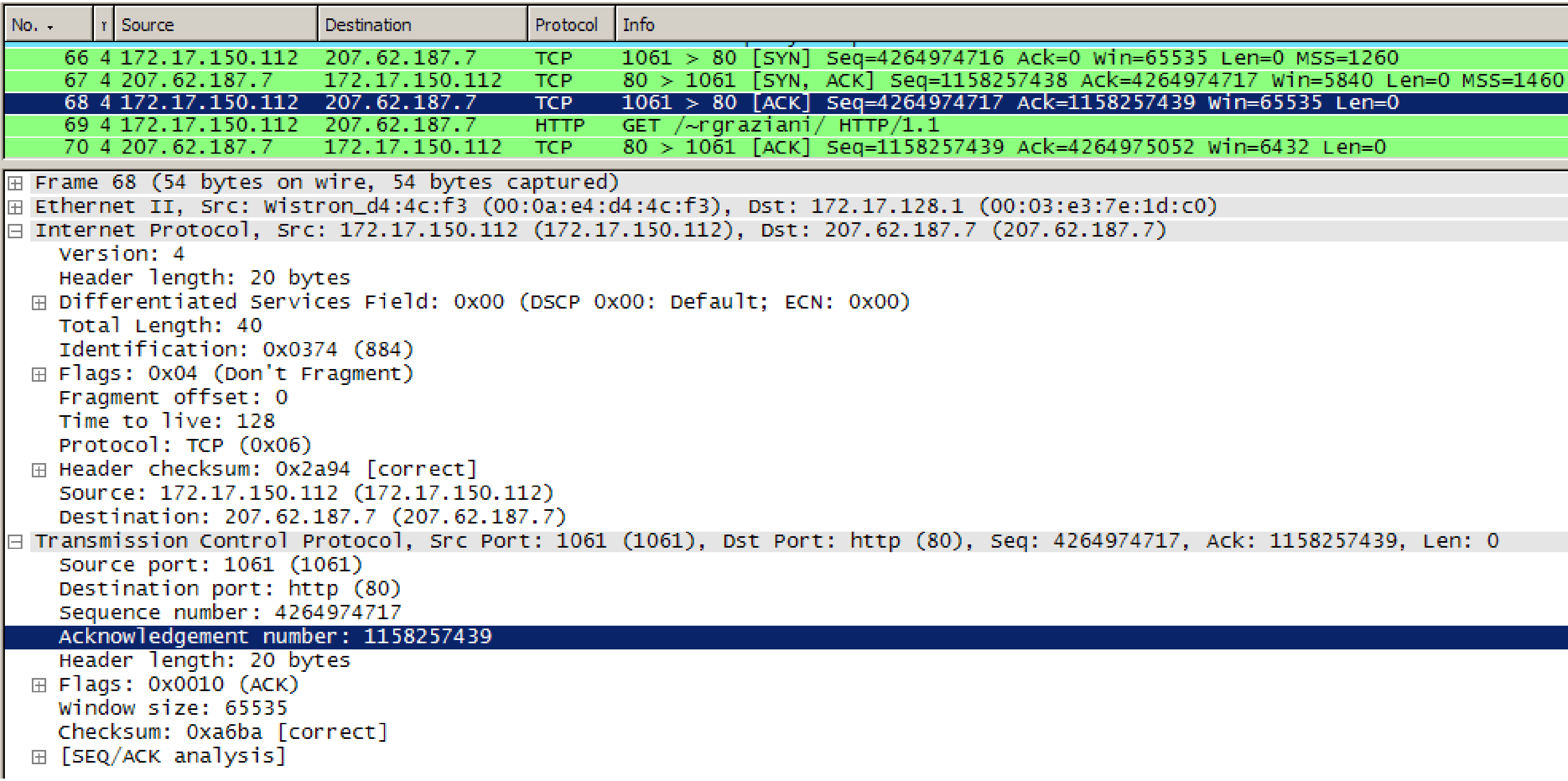
Server: ACK = 4264974717
Seq = 1158257438



```
No. .  r  Source          Destination     Protocol  Info
66     4  172.17.150.112  207.62.187.7    TCP       1061 > 80 [SYN] Seq=4264974716 Ack=0 Win=65535 Len=0 MSS=1260
67     4  207.62.187.7    172.17.150.112  TCP       80 > 1061 [SYN, ACK] Seq=1158257438 Ack=4264974717 Win=5840 Len=0 MSS=1460
68     4  172.17.150.112  207.62.187.7    TCP       1061 > 80 [ACK] Seq=4264974717 Ack=1158257439 Win=65535 Len=0
69     4  172.17.150.112  207.62.187.7    HTTP      GET /~rgraziani/ HTTP/1.1
70     4  207.62.187.7    172.17.150.112  TCP       80 > 1061 [ACK] Seq=1158257439 Ack=4264975052 Win=6432 Len=0
```

```
⊞ Frame 67 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: 172.17.128.1 (00:03:e3:7e:1d:c0), Dst: Wistron_d4:4c:f3 (00:0a:e4:d4:4c:f3)
⊟ Internet Protocol, Src: 207.62.187.7 (207.62.187.7), Dst: 172.17.150.112 (172.17.150.112)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 48
    Identification: 0x0000 (0)
  ⊞ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 62
    Protocol: TCP (0x06)
  ⊞ Header checksum: 0x7000 [correct]
    Source: 207.62.187.7 (207.62.187.7)
    Destination: 172.17.150.112 (172.17.150.112)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 1061 (1061), Seq: 1158257438, Ack: 4264974717, Len: 0
    Source port: http (80)
    Destination port: 1061 (1061)
    Sequence number: 1158257438
    Acknowledgement number: 4264974717
    Header length: 28 bytes
  ⊞ Flags: 0x0012 (SYN, ACK)
    Window size: 5840
    Checksum: 0x6326 [correct]
  ⊞ Options: (8 bytes)
  ⊞ [SEQ/ACK analysis]
```

Client: ACK = 1158257439



```
No.   T Source          Destination     Protocol Info
   66 4 172.17.150.112  207.62.187.7    TCP      1061 > 80 [SYN] Seq=4264974716 Ack=0 Win=65535 Len=0 MSS=1260
   67 4 207.62.187.7    172.17.150.112  TCP      80 > 1061 [SYN, ACK] Seq=1158257438 Ack=4264974717 Win=5840 Len=0 MSS=1460
   68 4 172.17.150.112  207.62.187.7    TCP      1061 > 80 [ACK] Seq=4264974717 Ack=1158257439 Win=65535 Len=0
   69 4 172.17.150.112  207.62.187.7    HTTP     GET /~rgraziani/ HTTP/1.1
   70 4 207.62.187.7    172.17.150.112  TCP      80 > 1061 [ACK] Seq=1158257439 Ack=4264975052 Win=6432 Len=0
```

```
⊞ Frame 68 (54 bytes on wire, 54 bytes captured)
⊞ Ethernet II, Src: Wistron_d4:4c:f3 (00:0a:e4:d4:4c:f3), Dst: 172.17.128.1 (00:03:e3:7e:1d:c0)
⊟ Internet Protocol, Src: 172.17.150.112 (172.17.150.112), Dst: 207.62.187.7 (207.62.187.7)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 40
    Identification: 0x0374 (884)
  ⊞ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
  ⊞ Header checksum: 0x2a94 [correct]
    Source: 172.17.150.112 (172.17.150.112)
    Destination: 207.62.187.7 (207.62.187.7)
⊟ Transmission Control Protocol, Src Port: 1061 (1061), Dst Port: http (80), Seq: 4264974717, Ack: 1158257439, Len: 0
    Source port: 1061 (1061)
    Destination port: http (80)
    Sequence number: 4264974717
    Acknowledgement number: 1158257439
    Header length: 20 bytes
  ⊞ Flags: 0x0010 (ACK)
    Window size: 65535
    Checksum: 0xa6ba [correct]
  ⊞ [SEQ/ACK analysis]
```

Another example

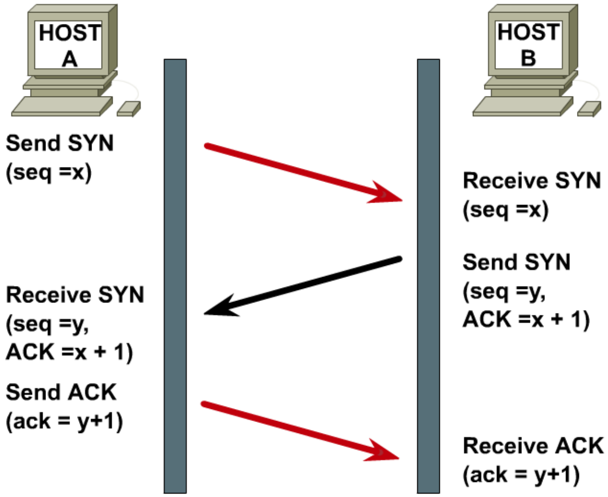
```
 Packet 1: source: 130.57.20.10  dest.:130.57.20.1
TCP: ----- TCP header -----
      TCP: Source port              = 1026
      TCP: Destination port         = 524
      TCP: Initial sequence number = 12952
      TCP: Next expected Seq number= 12953
      TCP:                .... ..1. = SYN
      TCP: Window                   = 8192
      TCP: Checksum                 = 1303 (correct)
      TCP: Maximum segment size     = 1460 (TCP Option)
```

```
Packet 2: source: 130.57.20.1   dest: 130.57.20.10
TCP: ----- TCP header -----
      TCP: Source port              = 524
      TCP: Destination port         = 1026
      TCP: Initial sequence number = 2744080
      TCP: Next expected Seq number= 2744081
      TCP: Acknowledgment number    = 12953
      TCP:                .... ..1. = SYN
      TCP: Window                   = 32768
      TCP: Checksum                 = D3B7 (correct)
      TCP: Maximum segment size     = 1460 (TCP Option)
```

```
Packet 3: source: 130.57.20.10  dest: 130.57.20.1
TCP: ----- TCP header -----
      TCP: Source port              = 1026
      TCP: Destination port         = 524
      TCP: Sequence number          = 12953
      TCP: Next expected Seq number= 12953
      TCP: Acknowledgment number    = 2744081
      TCP:                ...1 .... = Acknowledgment
      TCP: Window                   = 8760
      TCP: Checksum                 = 493D (correct)
      TCP: No TCP options
```

*

TCP Three-Way Handshake/ Open Connection



- Only part of the TCP headers are displayed.
- Notice that the Maximum segment size and the negotiated Window size are also sent.

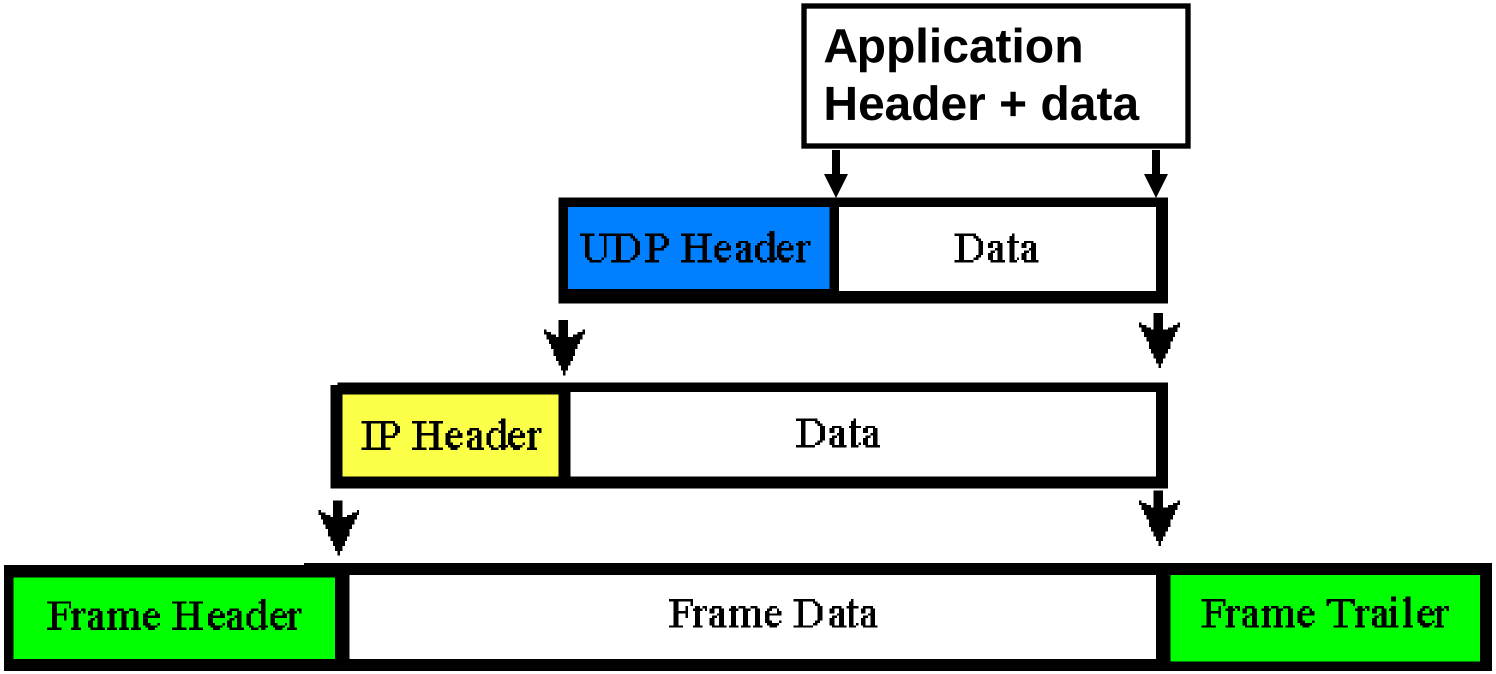
Topics

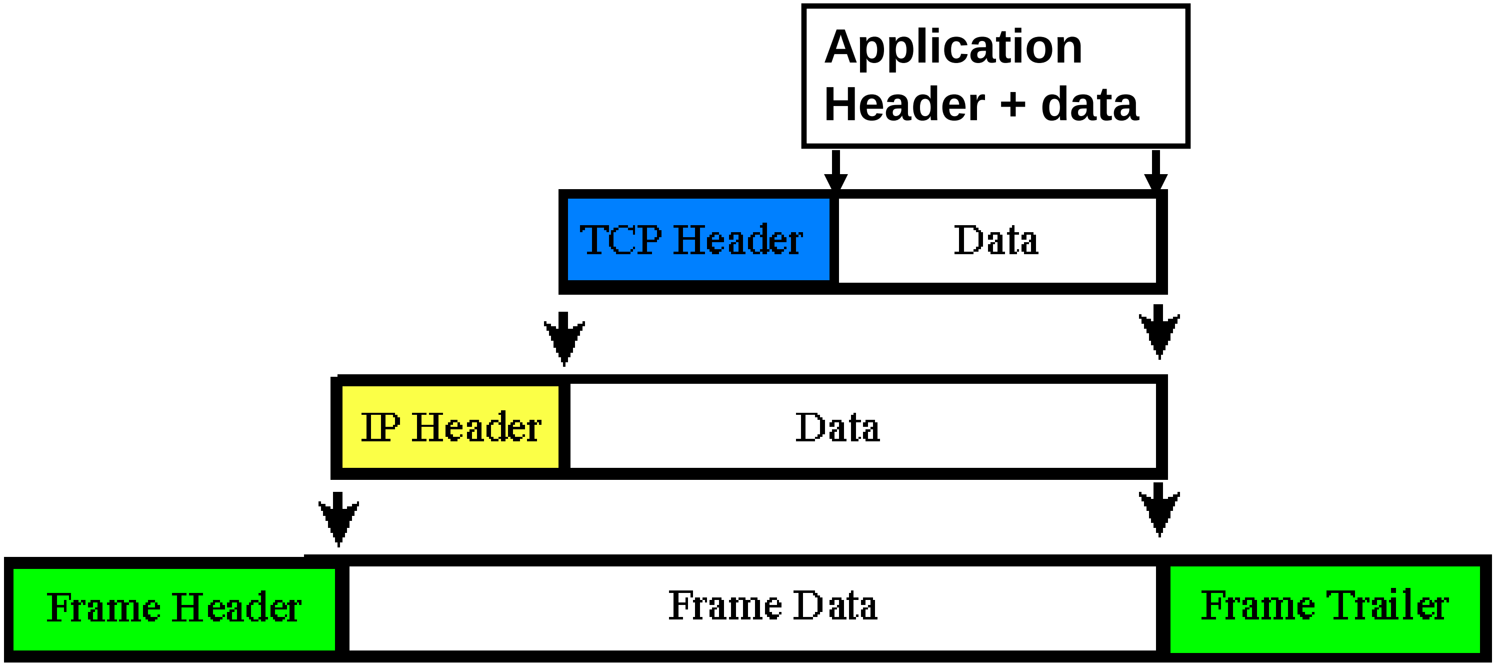
Layer 3 Concepts

- TCP/IP and the Internet Layer
- Diagram the IP datagram
- Internet Control Message Protocol (ICMP)

TCP/IP protocol stack and the transport layer

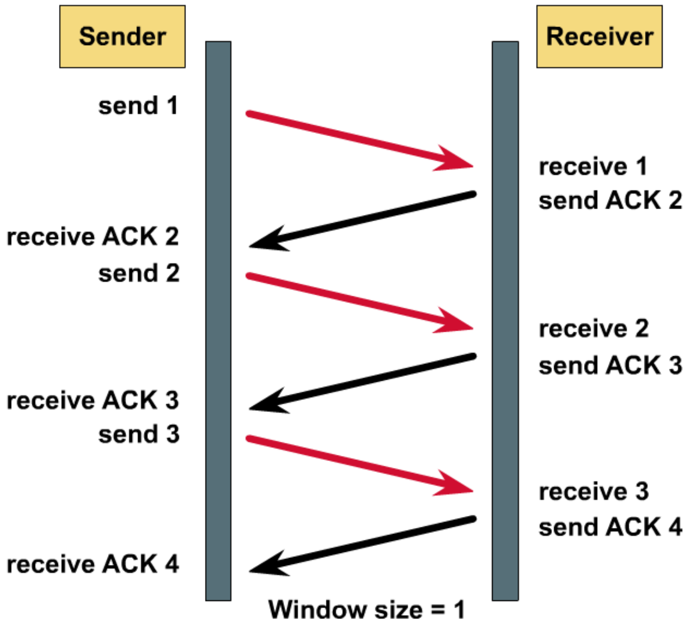
- TCP and UDP segment format
- TCP and UDP port numbers
- TCP three-way handshake/open connection
- TCP simple acknowledgment and windowing


Application
Header + data
UDP Header
Data
IP Header
Data
Frame Header
Frame Data
Frame Trailer


Application
Header + data
TCP Header
Data
IP Header
Data
Frame Header
Frame Data
Frame Trailer

TCP Windows and Sliding Windows

TCP Simple Acknowledgment



Sender

Receiver

send 1

receive 1
send ACK 2

receive ACK 2
send 2

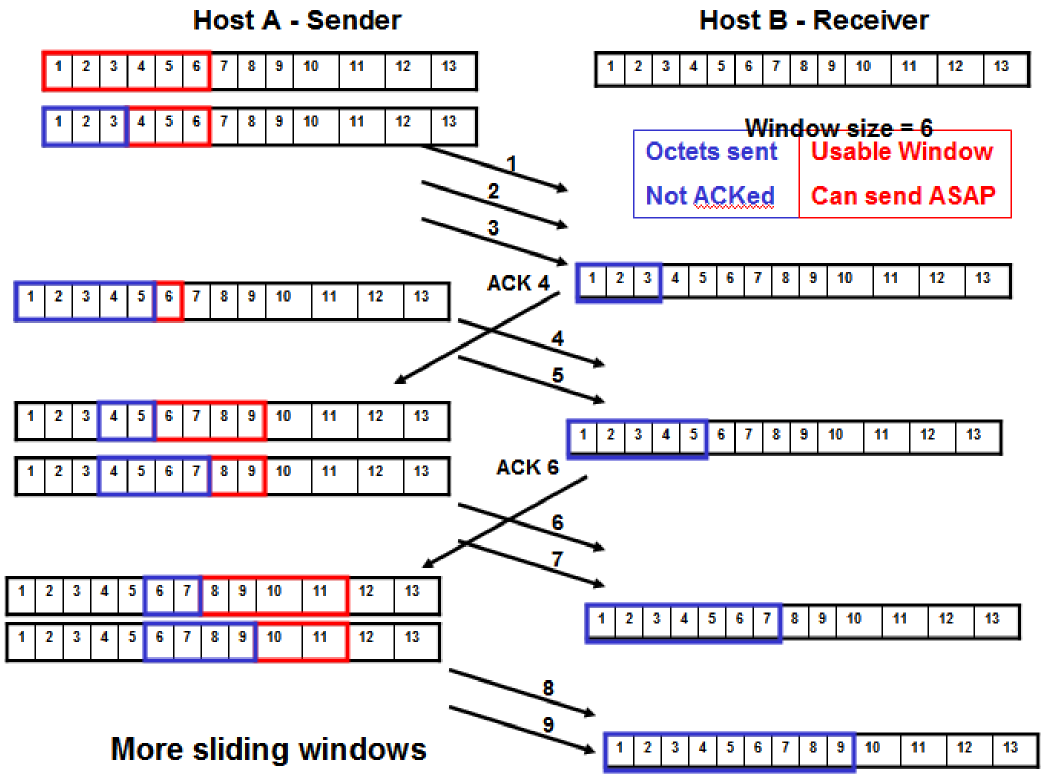
receive 2
send ACK 3

receive ACK 3
send 3

receive 3
send ACK 4

receive ACK 4

Window size = 1



Host A - Sender

Host B - Receiver

Window size = 6

Octets sent Not ACKed

Usable Window Can send ASAP

1

2

3

ACK 4

4

5

ACK 6

6

7

8

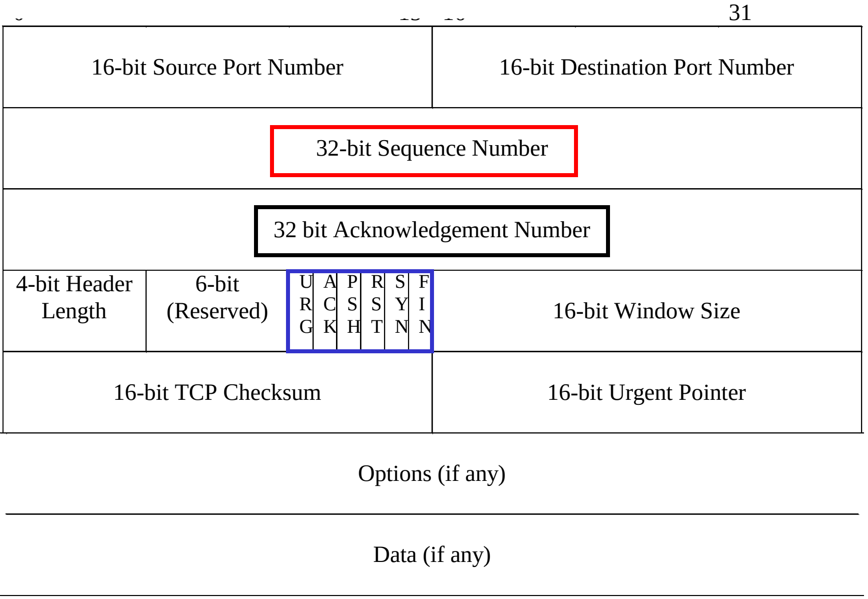
9

More sliding windows

Over Simplification

- Note: The following examples of Window Size, Sliding Windows, and Retransmission are very simplistic examples using 1 byte segments. This is meant only to introduce the reader to TCP and is not intended to give realistic examples.

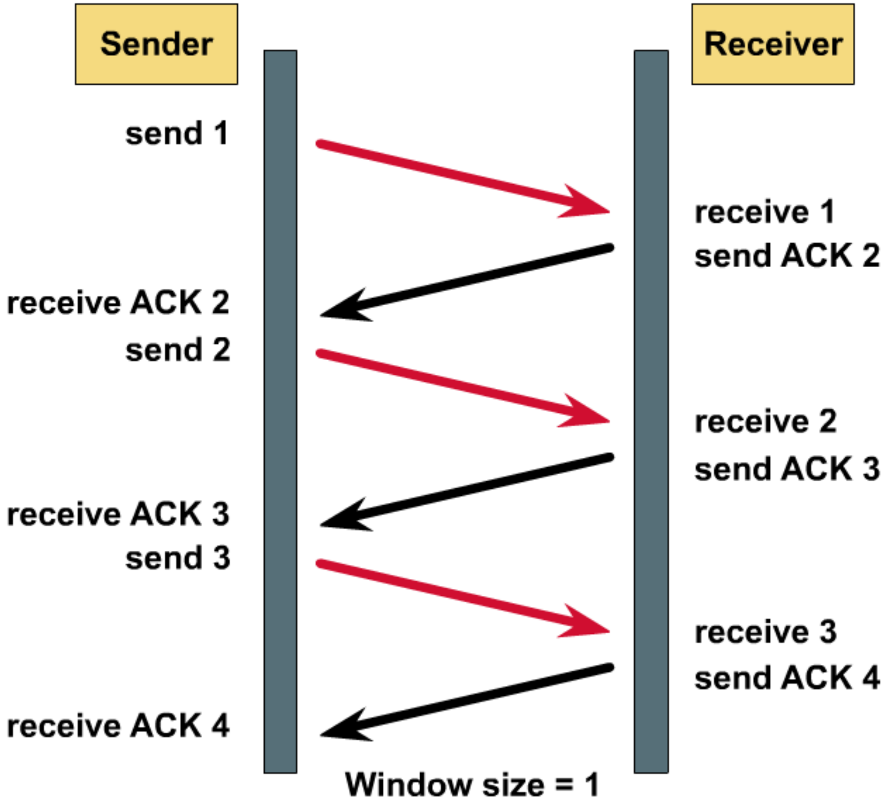
TCP Header



31

<table>
<tr><td colspan="8">16-bit Source Port Number</td><td>16-bit Destination Port Number</td></tr>
<tr><td colspan="9">32-bit Sequence Number</td></tr>
<tr><td colspan="9">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16-bit Window Size</td></tr>
<tr><td colspan="8">16-bit TCP Checksum</td><td>16-bit Urgent Pointer</td></tr>
<tr><td colspan="9">Options (if any)</td></tr>
<tr><td colspan="9">Data (if any)</td></tr>
</table>

TCP Simple Acknowledgment



Flow Control and Reliability

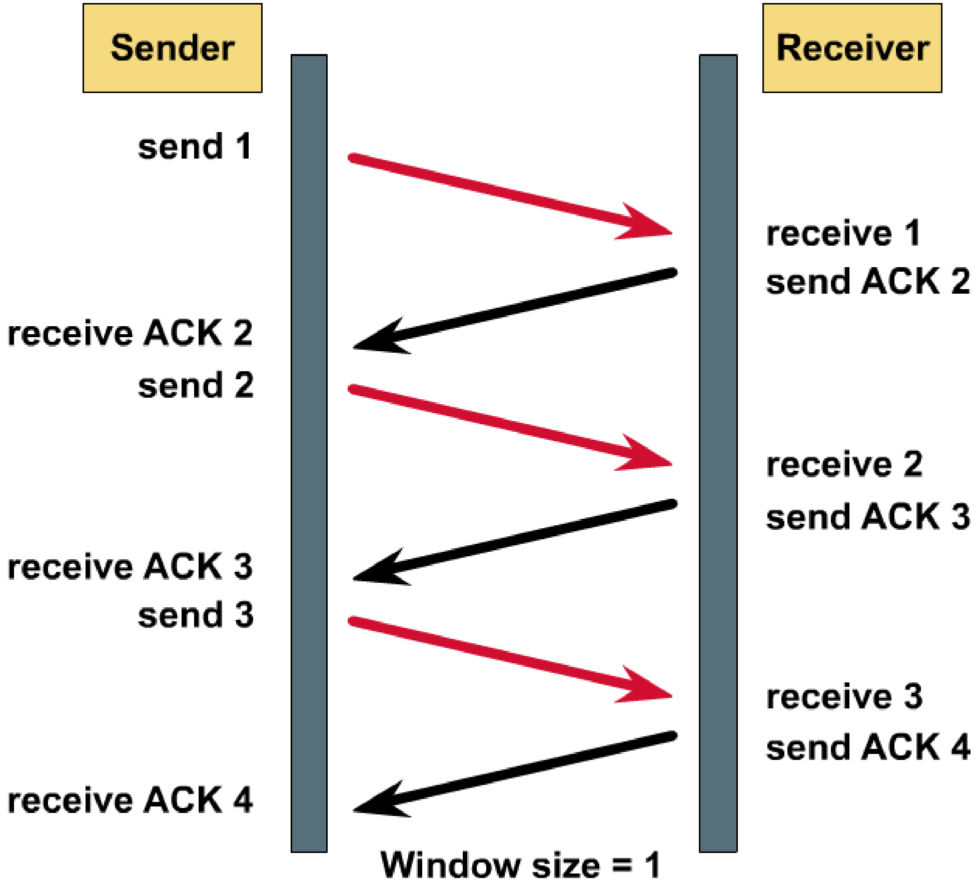
- To govern the flow of data between devices, TCP uses a peer-to-peer flow control mechanism.
- The receiving host's TCP layer reports a window size to the sending host's TCP layer.
- This **window size** specifies the number of bytes, starting with the acknowledgment number, that the receiving host's TCP layer is currently prepared to receive.

TCP Header

31

<table>
<tr><td colspan="8">16-bit Source Port Number</td><td>16-bit Destination Port Number</td></tr>
<tr><td colspan="9">32-bit Sequence Number</td></tr>
<tr><td colspan="9">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16-bit Window Size</td></tr>
<tr><td colspan="8">16-bit TCP Checksum</td><td>16-bit Urgent Pointer</td></tr>
<tr><td colspan="9">Options (if any)</td></tr>
<tr><td colspan="9">Data (if any)</td></tr>
</table>

TCP Simple Acknowledgment



Sender

Receiver

send 1

receive 1

send ACK 2

receive ACK 2

send 2

receive 2

send ACK 3

receive ACK 3

send 3

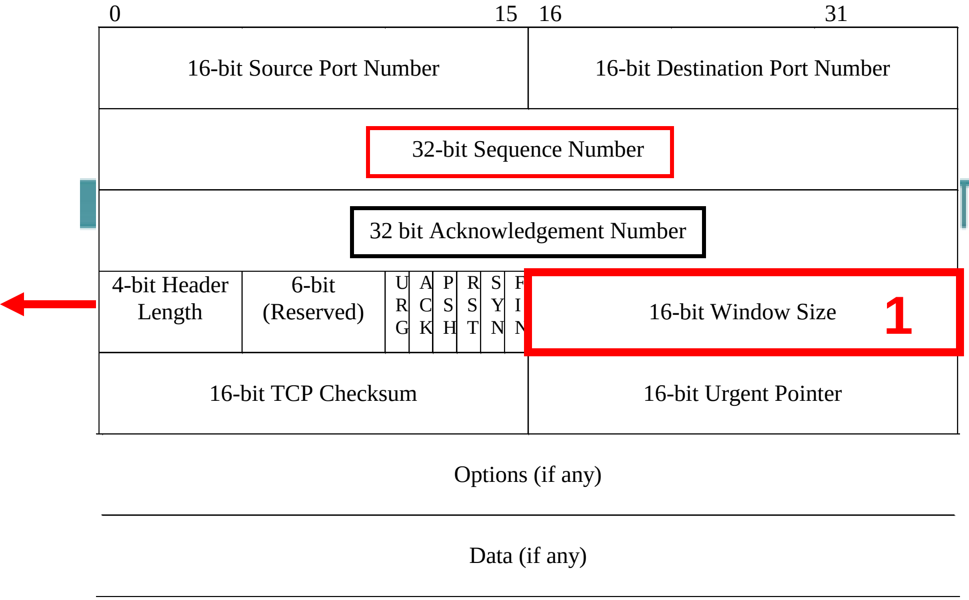
receive 3

send ACK 4

receive ACK 4

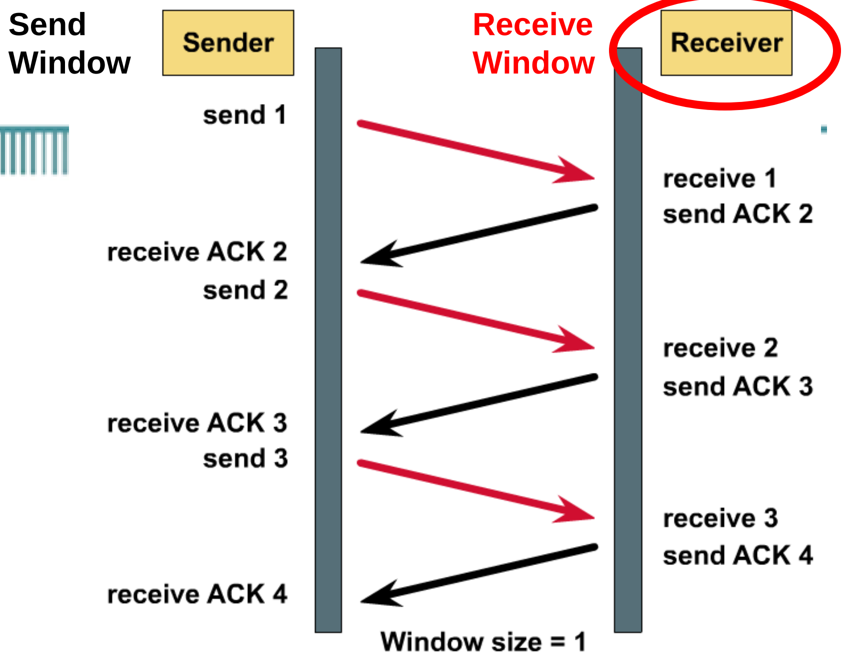
Window size = 1

- **TCP** -- a connection-oriented, reliable protocol; provides **flow control** by providing sliding windows, and **reliability** by providing sequence numbers and acknowledgments.
- TCP re-sends anything that is not received and supplies a “TCP” **virtual circuit** between end-user applications.
- The advantage of TCP is that it provides **guaranteed delivery** of the segments.



0	15	16	31
16-bit Source Port Number		16-bit Destination Port Number	
32-bit Sequence Number			
32 bit Acknowledgement Number			
4-bit Header Length	6-bit (Reserved)	U A P R S F R C S S Y I G K H T N N	16-bit Window Size 1
16-bit TCP Checksum		16-bit Urgent Pointer	
Options (if any)			
Data (if any)			

TCP Simple Acknowledgment



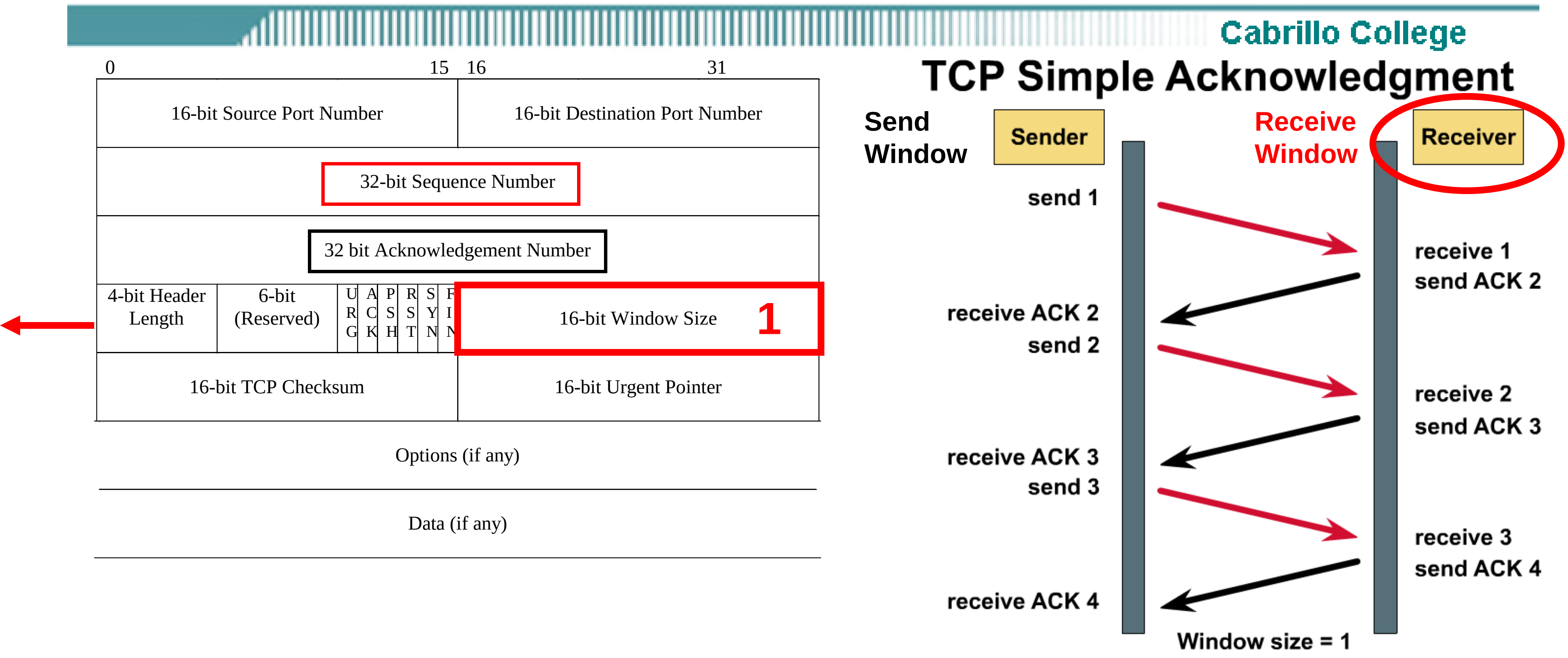
Receive Window

- The TCP Receive Window size is the amount of receive data (in bytes) that can be buffered by this host, at one time on a connection.
- The other (sending) host can send only that amount of data before getting an acknowledgment and window update from this (the receiving) host.

Send Window (not a TCP field)

- The TCP Receive Window size of the other host.
- How much data (in bytes) that can be sent by this host before receiving an acknowledgement from the other host.

TCP Window Size

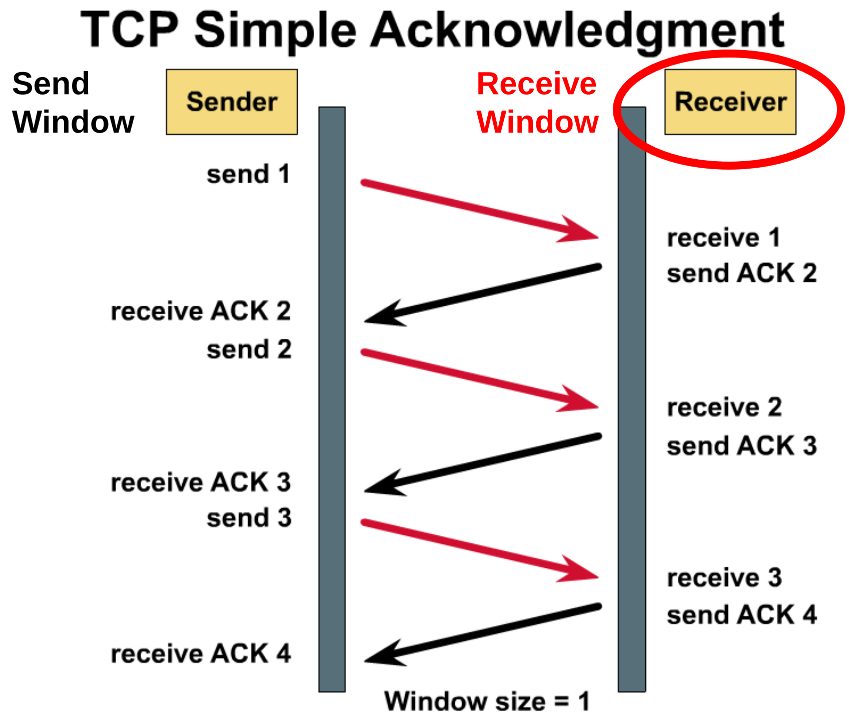


- After a host transmits the window-size number of bytes, it must receive an acknowledgment before any more data can be sent.
- The window size determines how much data the receiving station can accept at one time.

TCP Window Size

Cabrillo College

<table>
<tr><td colspan="8">0   15</td><td>16   31</td></tr>
<tr><td colspan="8">16-bit Source Port Number</td><td>16-bit Destination Port Number</td></tr>
<tr><td colspan="9">32-bit Sequence Number</td></tr>
<tr><td colspan="9">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td>16-bit Window Size 1</td></tr>
<tr><td colspan="8">16-bit TCP Checksum</td><td>16-bit Urgent Pointer</td></tr>
<tr><td colspan="9">Options (if any)</td></tr>
<tr><td colspan="9">Data (if any)</td></tr>
</table>



TCP Simple Acknowledgment

Send Window

Sender

Receive Window

Receiver

send 1

receive 1

send ACK 2

receive ACK 2

send 2

receive 2

send ACK 3

receive ACK 3

send 3

receive 3

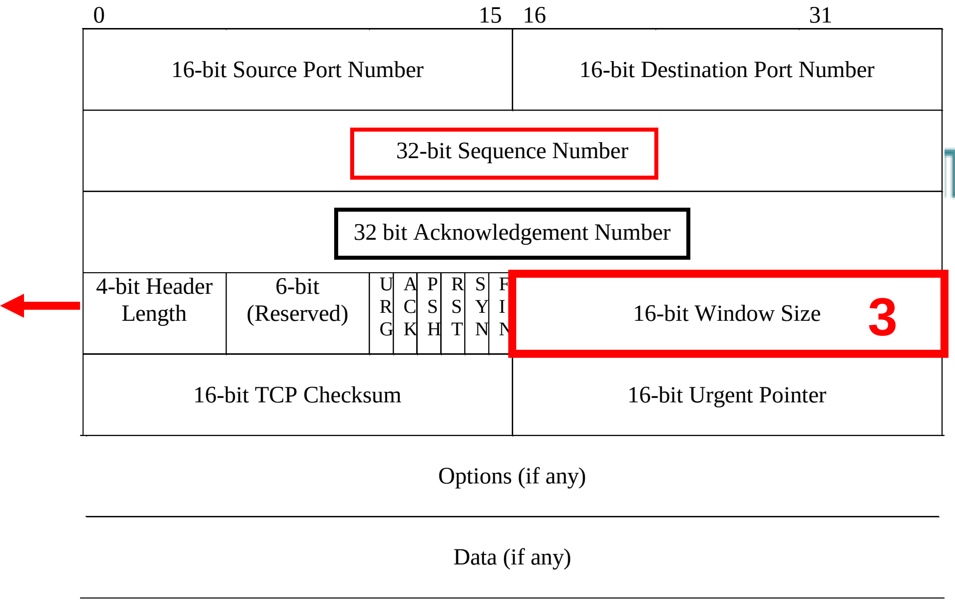
send ACK 4

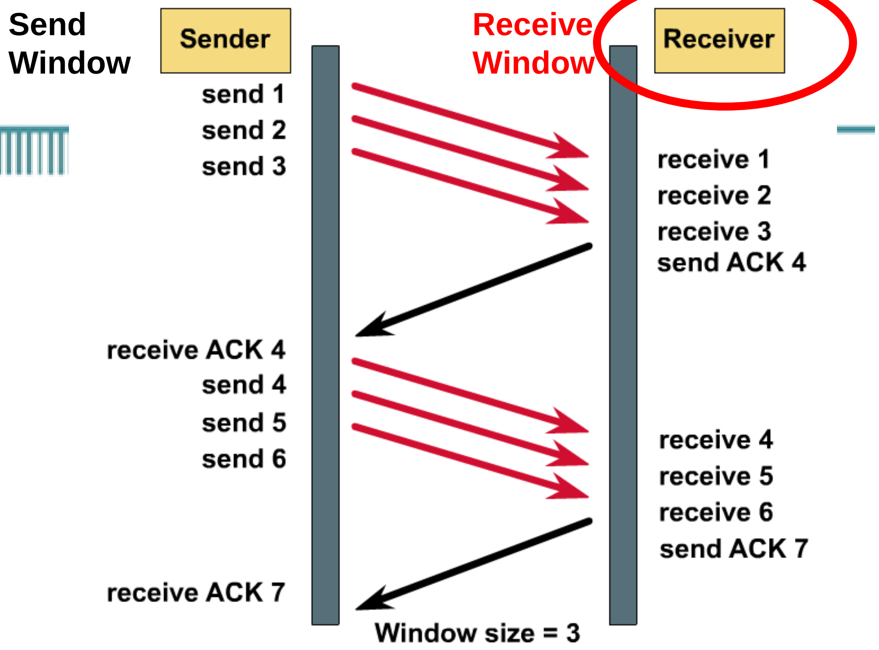
receive ACK 4

Window size = 1

- With a **window size of 1**, each segment carries only one byte of data and must be acknowledged before another segment is transmitted.
- This results in **inefficient** host use of bandwidth.
- The purpose of windowing is to improve flow control and reliability.
- Unfortunately, with a window size of 1, you see a very inefficient use of bandwidth.

TCP ~~Sliding~~ Window



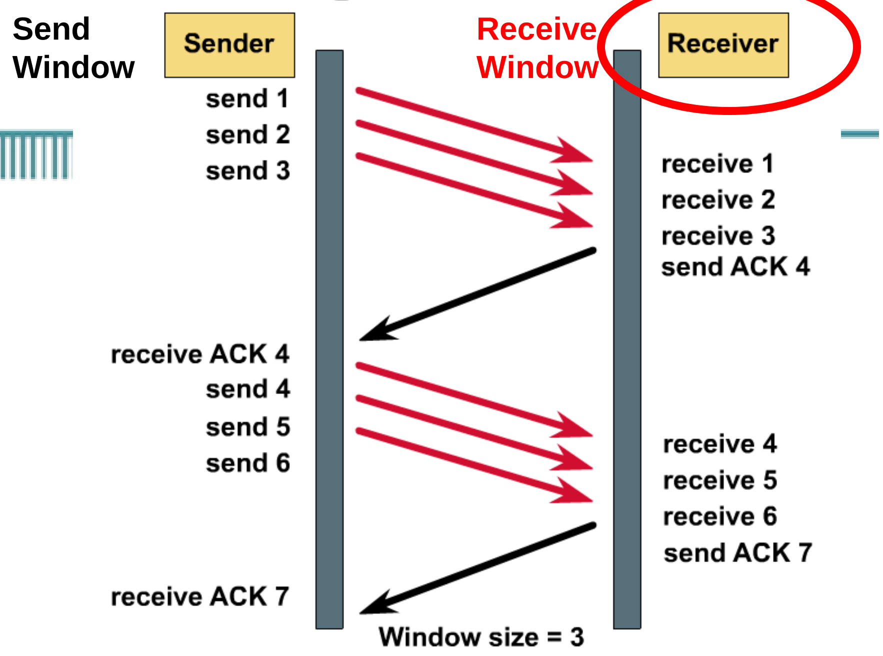


Receiver’s TCP Window Size

- TCP uses a window size, number of bytes, that the receiver is willing to accept, and is usually controlled by the receiving process.
- TCP uses **expectational acknowledgments**, meaning that the acknowledgment number refers to the next byte that the sender of the acknowledgement expects to receive.
- A larger window size allows more data to be transmitted pending acknowledgment.
- Note: The sequence number being sent identifies the first byte of data in that segment.

<table>
<tr><th colspan="3">0</th><th>15</th><th>16</th><th>31</th></tr>
<tr><td colspan="4">16-bit Source Port Number</td><td colspan="2">16-bit Destination Port Number</td></tr>
<tr><td colspan="6">32-bit Sequence Number</td></tr>
<tr><td colspan="6">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td colspan="2">URG | ACK | PSH | RST | SYN | FIN</td><td colspan="2">16-bit Window Size 3</td></tr>
<tr><td colspan="4">16-bit TCP Checksum</td><td colspan="2">16-bit Urgent Pointer</td></tr>
<tr><td colspan="6">Options (if any)</td></tr>
<tr><td colspan="6">Data (if any)</td></tr>
</table>

TCP ~~Sliding~~ Window



TCP Window Size

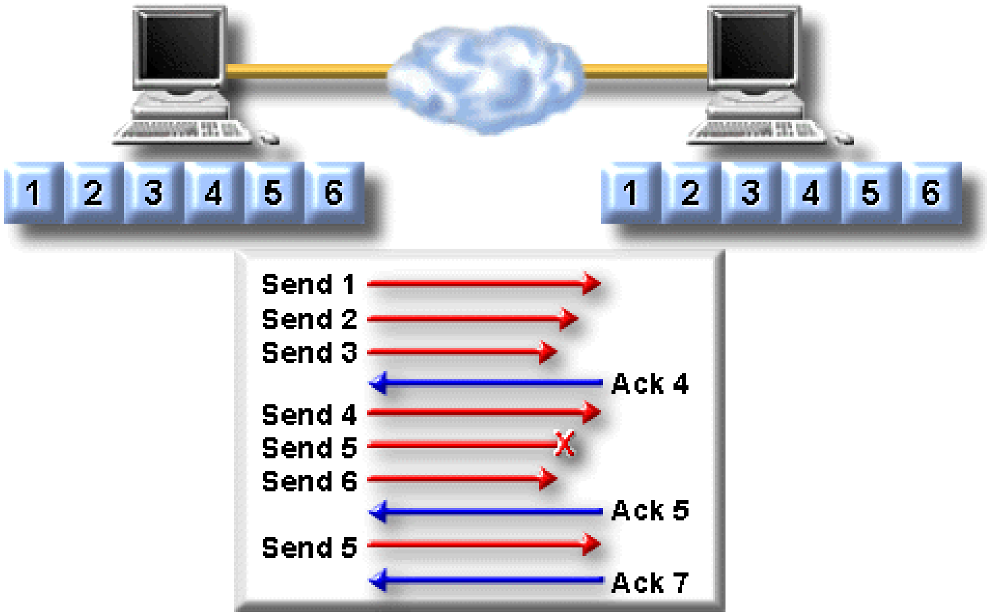
- TCP provides **full-duplex service**, which means data can be flowing in each direction, independent of the other direction.
- Window sizes, sequence numbers and acknowledgment numbers are independent of each other’s data flow.
- Receiver sends acceptable window size to sender during each segment transmission (flow control)
  - if too much data being sent, acceptable window size is reduced
  - if more data can be handled, acceptable window size is increased
- This is known as a **Stop-and-Wait** windowing protocol.

<table>
<tr><td colspan="4">0</td><td>15</td><td>16</td><td>31</td></tr>
<tr><td colspan="5">16-bit Source Port Number</td><td colspan="2">16-bit Destination Port Number</td></tr>
<tr><td colspan="7">32-bit Sequence Number</td></tr>
<tr><td colspan="7">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>U
R
G</td><td>A
C
K</td><td>P S H | R S T</td><td>S Y N | F I N</td><td>16-bit Window Size</td></tr>
<tr><td colspan="5">16-bit TCP Checksum</td><td colspan="2">16-bit Urgent Pointer</td></tr>
<tr><td colspan="7">Options (if any)</td></tr>
<tr><td colspan="7">Data (if any)</td></tr>
</table>

TCP Header

3

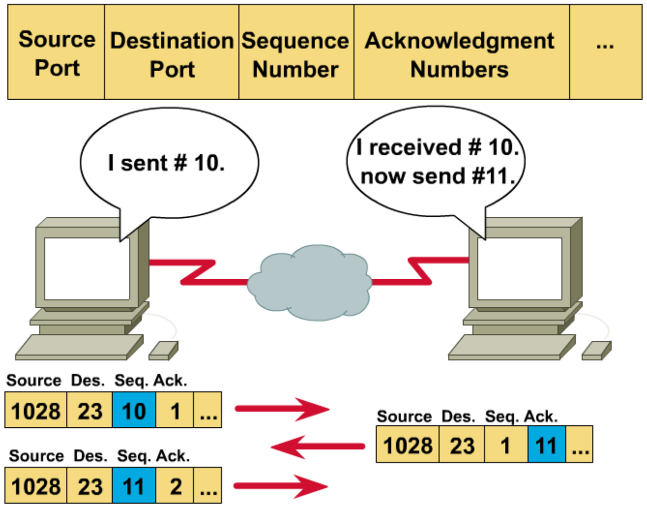
An Acknowledgment Technique



- Packets may be dropped along the way, timed out, or corrupted.
- If octets 4, 5, and 6 were sent, but 5 was lost, the receiver would only acknowledge up to 4, sending an Ack of 5.
- The sender would send 5 and wait to hear from the receiver where it should start up again.
- The receiver sends Ack 7, so the sender knows it can start sending again with octet 7.
- *There are **selective acknowledgements (SACK**) – not discussed here, which is a way of acknowledging selected pieces of the data stream.*

TCP Sequence and Acknowledgment Numbers

This is only if one octet was sent at a time, but what if multiple bytes are sent , which is usually the case?



Source Port	Destination Port	Sequence Number	Acknowledgment Numbers	...

Source	Des.	Seq.	Ack.	
1028	23	10	1	...

Source	Des.	Seq.	Ack.	
1028	23	1	11	...

Source	Des.	Seq.	Ack.	
1028	23	11	2	...

Tech Note (FYI)

- Sender: The value in the sequence number is the first byte in the data stream.
- So, how does the receiver know how much data was sent, so it knows what value to send in the acknowledgement?
- Receiver: Using the sender's IP packet and TCP segment information, the value of the ACK is:

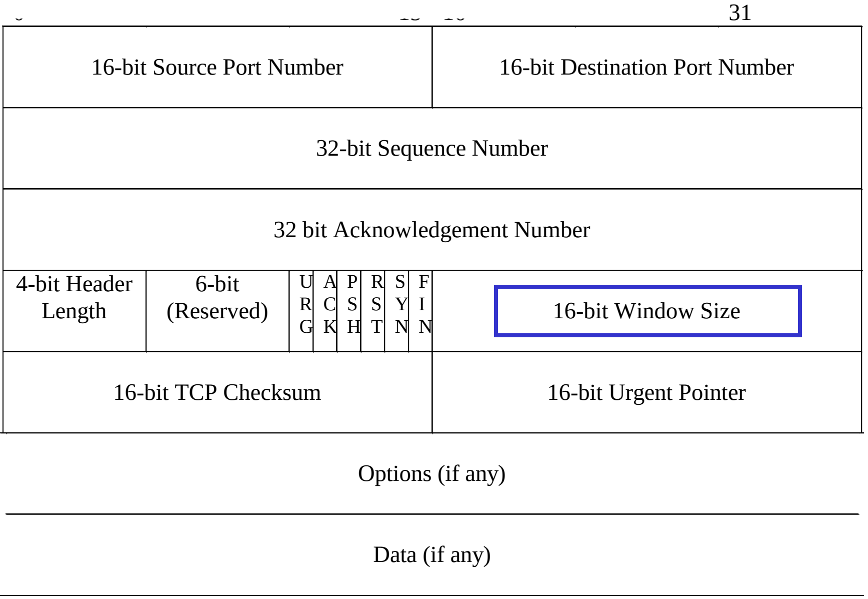
```
    IP Length: (IP header) Total length - Header length
 -  TCP header length (TCP header):  Header length
    -------------------------------------------------
    Length of data in TCP segment

    ACK = Last Sequence Number acked + Length of data in TCP segment
```

- Check Sequence Number to check for missing segments and to sequence out-of-order segments.
- Remember that the ACK is for the sequence number of the byte you expect to receive. When you ACK 101, that says you've received all bytes through 100. This ignores SACK.

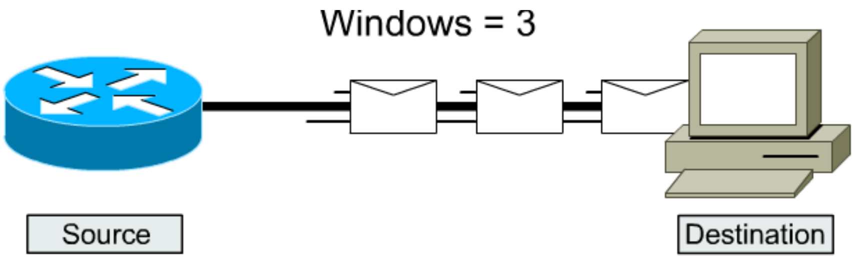
Sliding Windows

- Note: The following two slides on Sliding Windows contains corrections to the on-line curriculum followed by my slides on Sliding Windows.

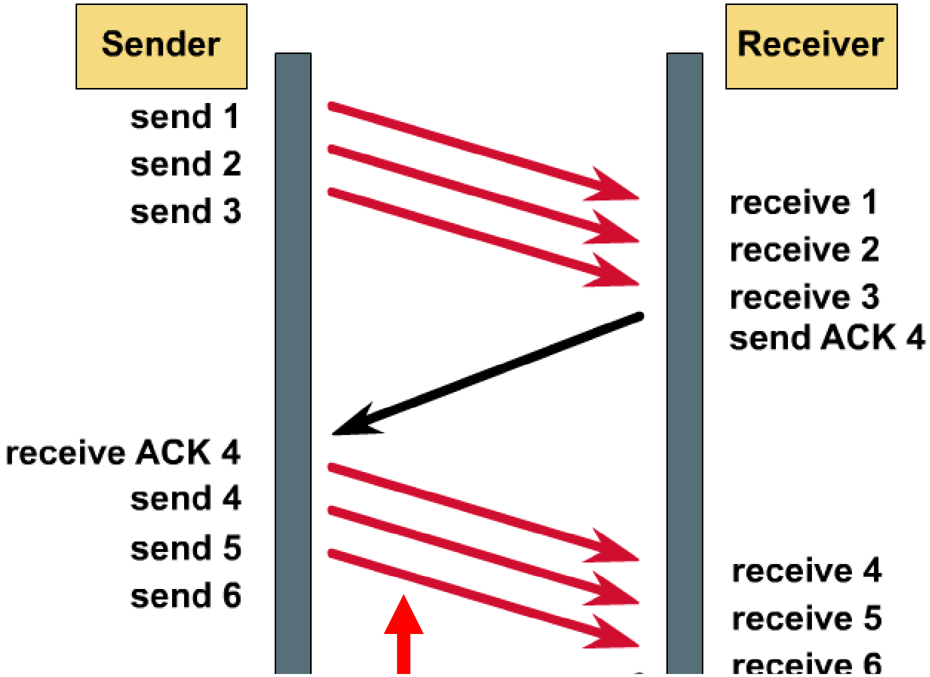
TCP Header



								31
16-bit Source Port Number								16-bit Destination Port Number
32-bit Sequence Number								
32 bit Acknowledgement Number								
4-bit Header Length	6-bit (Reserved)	URG	ACK	PSH	RST	SYN	FIN	16-bit Window Size
16-bit TCP Checksum								16-bit Urgent Pointer
Options (if any)								
Data (if any)								



TCP Sliding Window



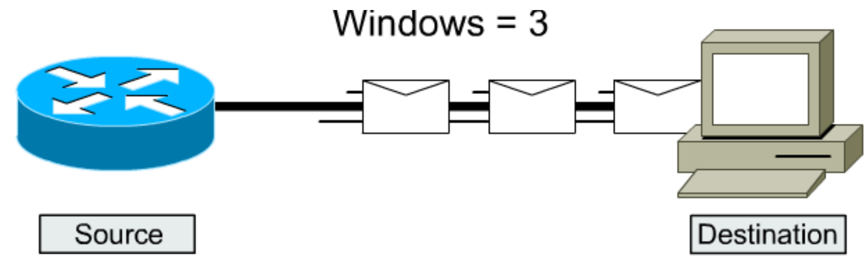
From Cisco Curriculum: This diagram is <u>not</u> an example of a sliding window, but of a window size of 3.

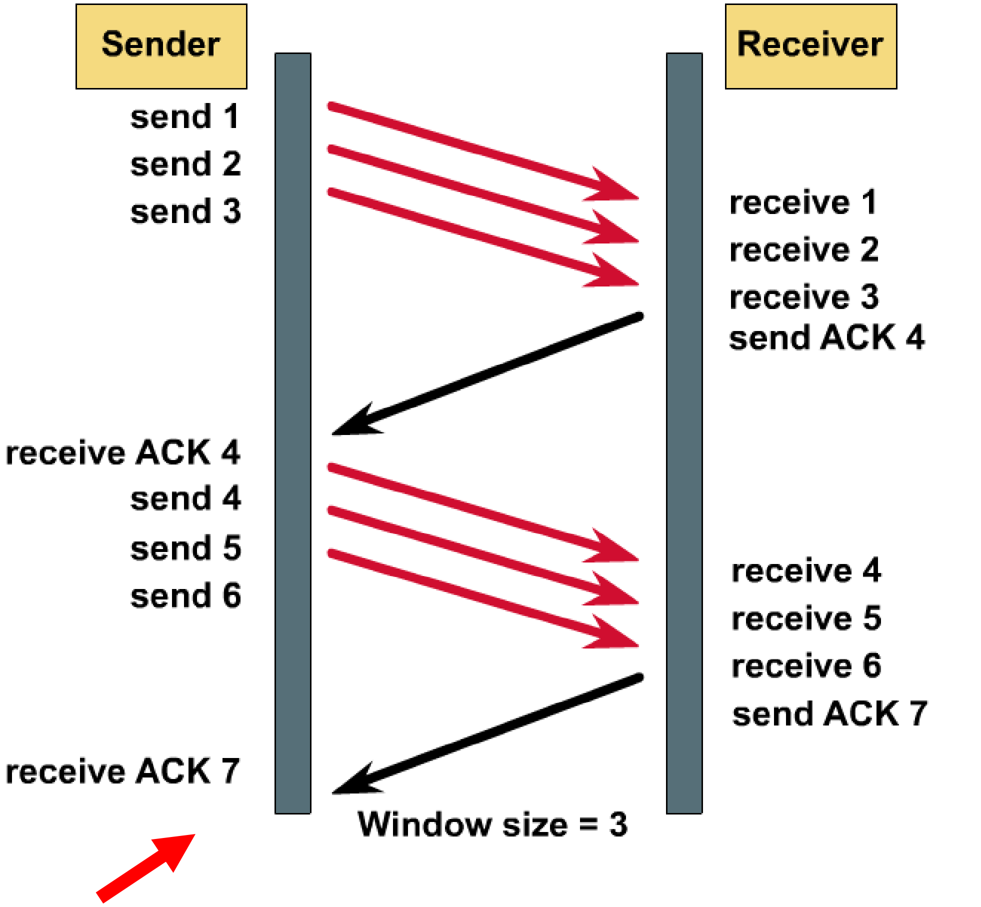
- TCP uses expectational acknowledgments, meaning that the acknowledgment number refers to the octet expected next.
- "The *sliding* part of *sliding window* refers to the fact that the window size is negotiated dynamically during the TCP session." **(This is <u>not exactly</u> what a sliding window is! Coming soon!)**
- A sliding window results in more efficient host use of bandwidth because a larger window size allows more data to be transmitted pending acknowledgment.

TCP Header

~~TCP Sliding~~ Window

<table>
<tr><th colspan="8"></th><th>31</th></tr>
<tr><td colspan="8">16-bit Source Port Number</td><td>16-bit Destination Port Number</td></tr>
<tr><td colspan="9">32-bit Sequence Number</td></tr>
<tr><td colspan="9">32 bit Acknowledgement Number</td></tr>
<tr><td>4-bit Header Length</td><td>6-bit (Reserved)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16-bit Window Size</td></tr>
<tr><td colspan="8">16-bit TCP Checksum</td><td>16-bit Urgent Pointer</td></tr>
<tr><td colspan="9">Options (if any)</td></tr>
<tr><td colspan="9">Data (if any)</td></tr>
</table>

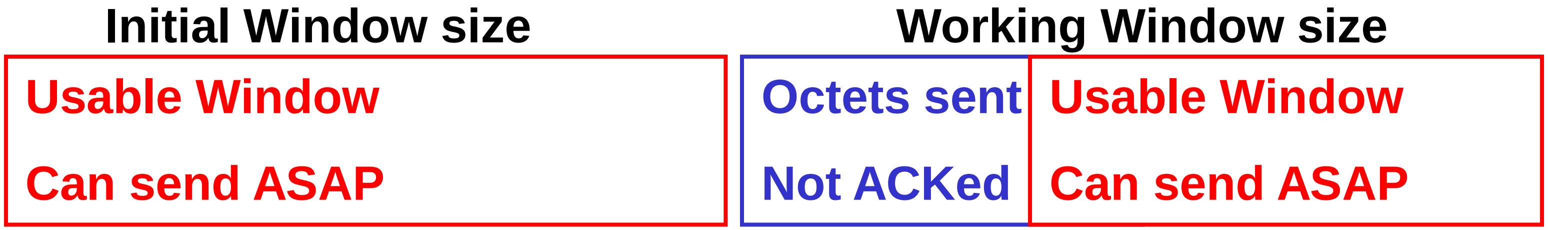




This diagram is <u>not</u> an example of a sliding window, but of a window size of 3.

- From Cisco Curriculum: “A sliding window results in more efficient host use of bandwidth because a larger window size allows more data to be transmitted pending acknowledgment. “ **(A larger window size does this, <u>not</u> a sliding window.)**

Sliding Windows



Initial Window size

Usable Window

Can send ASAP

Working Window size

Octets sent

Not ACKed

Usable Window

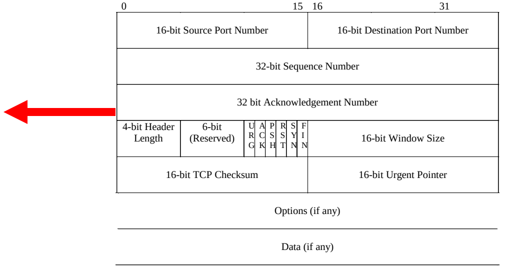
Can send ASAP

Sliding Window Protocol

- Sliding window algorithms are a method of flow control for network data transfers using the receivers Window size.
- The sender computes its usable window, which is how much data it can immediately send.
- Over time, this sliding window moves to the rights, as the receiver acknowledges data.
- The receiver sends acknowledgements as its TCP receive buffer empties.
- The terms used to describe the movement of the left and right edges of this sliding window are:

1. The left edge closes (moves to the right) when data is sent and acknowledged.
2. The right edge opens (moves to the right) allowing more data to be sent. This happens when the receiver acknowledges a certain number of bytes received.
3. The middle edge open (moves to the right) as data is sent, but not yet acknowledged.

TCP Header



0 15 16 31

16-bit Source Port Number

16-bit Destination Port Number

32-bit Sequence Number

32 bit Acknowledgement Number

4-bit Header Length

6-bit (Reserved)

URG ACK PSH RST SYN FIN

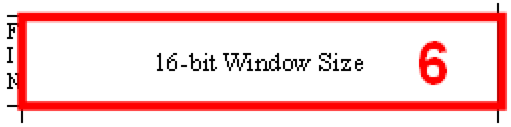
16-bit Window Size

16-bit TCP Checksum

16-bit Urgent Pointer

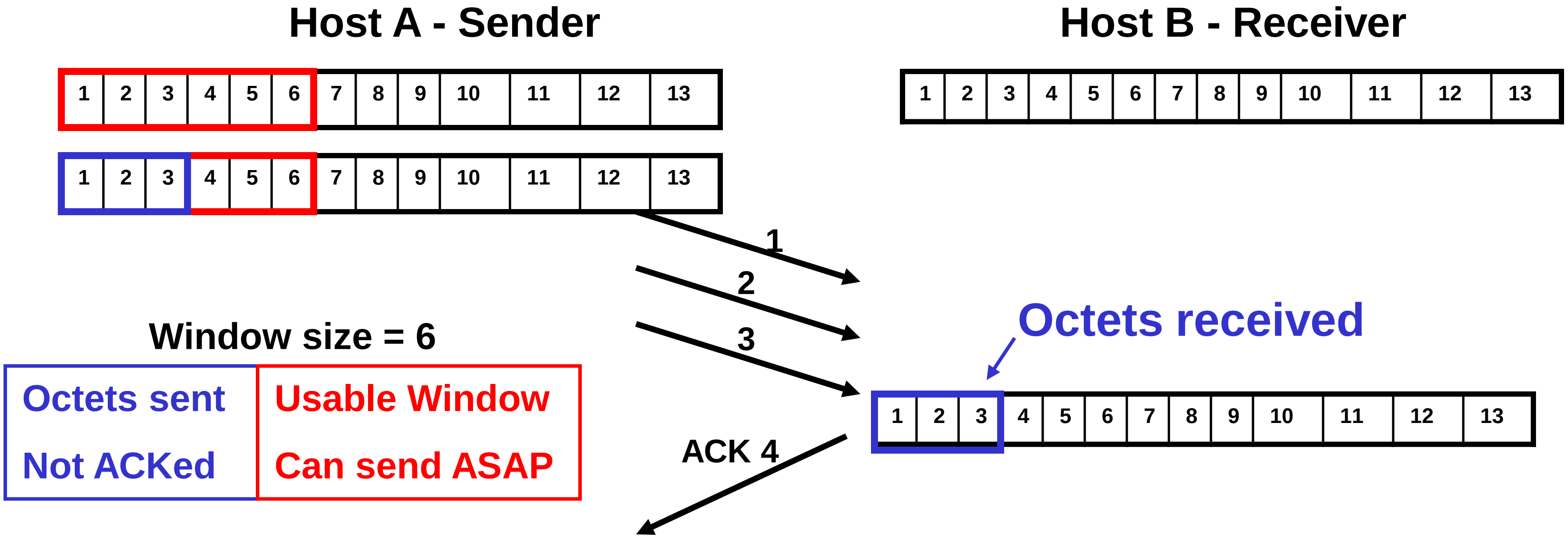
Options (if any)

Data (if any)



FIN

16-bit Window Size 6



Host A - Sender

Host B - Receiver

1 2 3 4 5 6 7 8 9 10 11 12 13

1 2 3 4 5 6 7 8 9 10 11 12 13

1 2 3 4 5 6 7 8 9 10 11 12 13

1 2 3

Window size = 6

Octets sent

Not ACKed

Usable Window

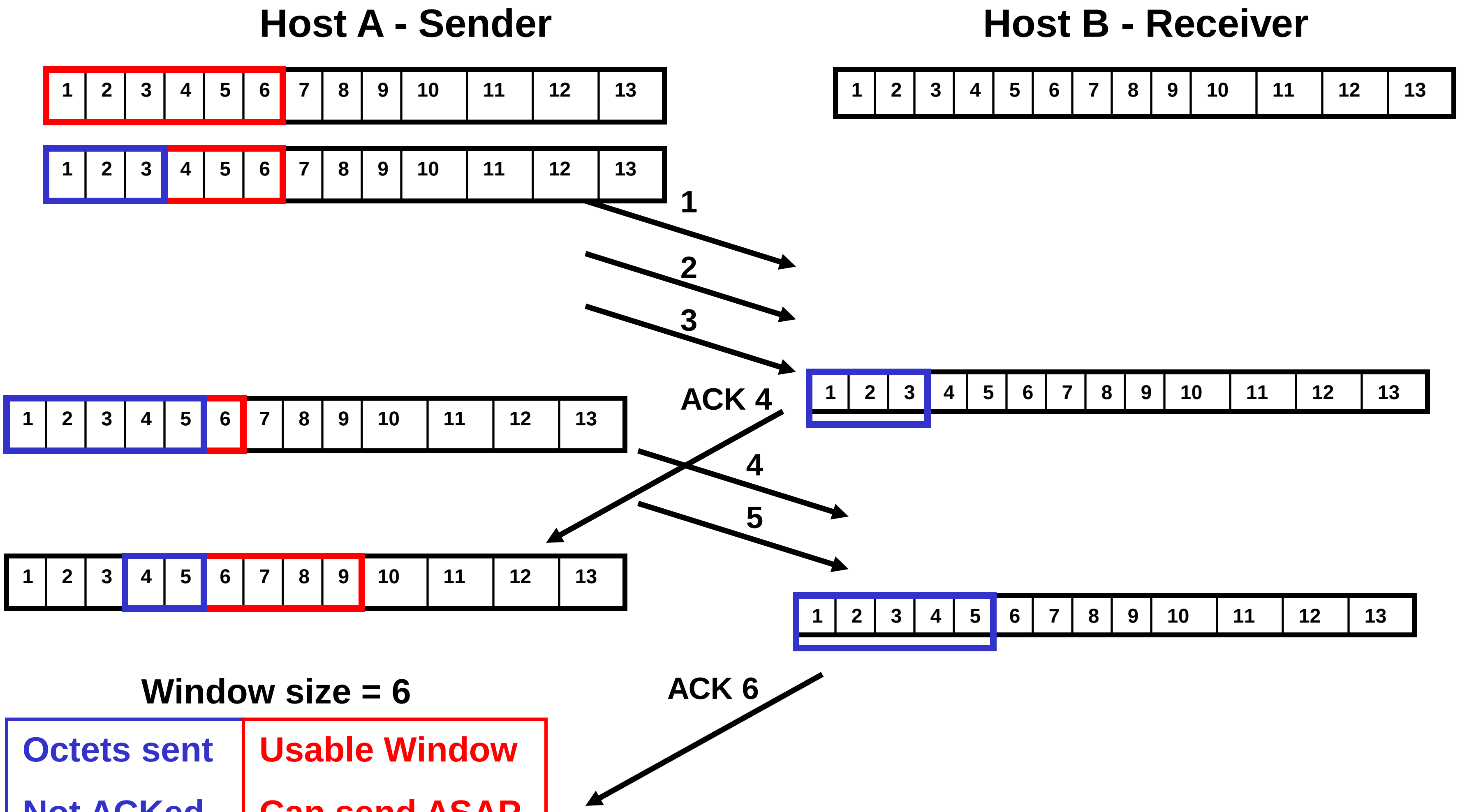
Can send ASAP

Octets received

1 2 3 4 5 6 7 8 9 10 11 12 13

ACK 4

- Host B gives Host A a window size of 6 (octets).
- Host A begins by sending octets to Host B: octets 1, 2, and 3 and slides it’s window over showing it has sent those 3 octets.
- Host A will not increase its usable window size by 3, until it receives an ACKnowldegement from Host B that it has received some or all of the octets.
- Host B, not waiting for all of the 6 octets to arrive, after receiving the third octet sends an expectational ACKnowledgement of “4” to Host A.



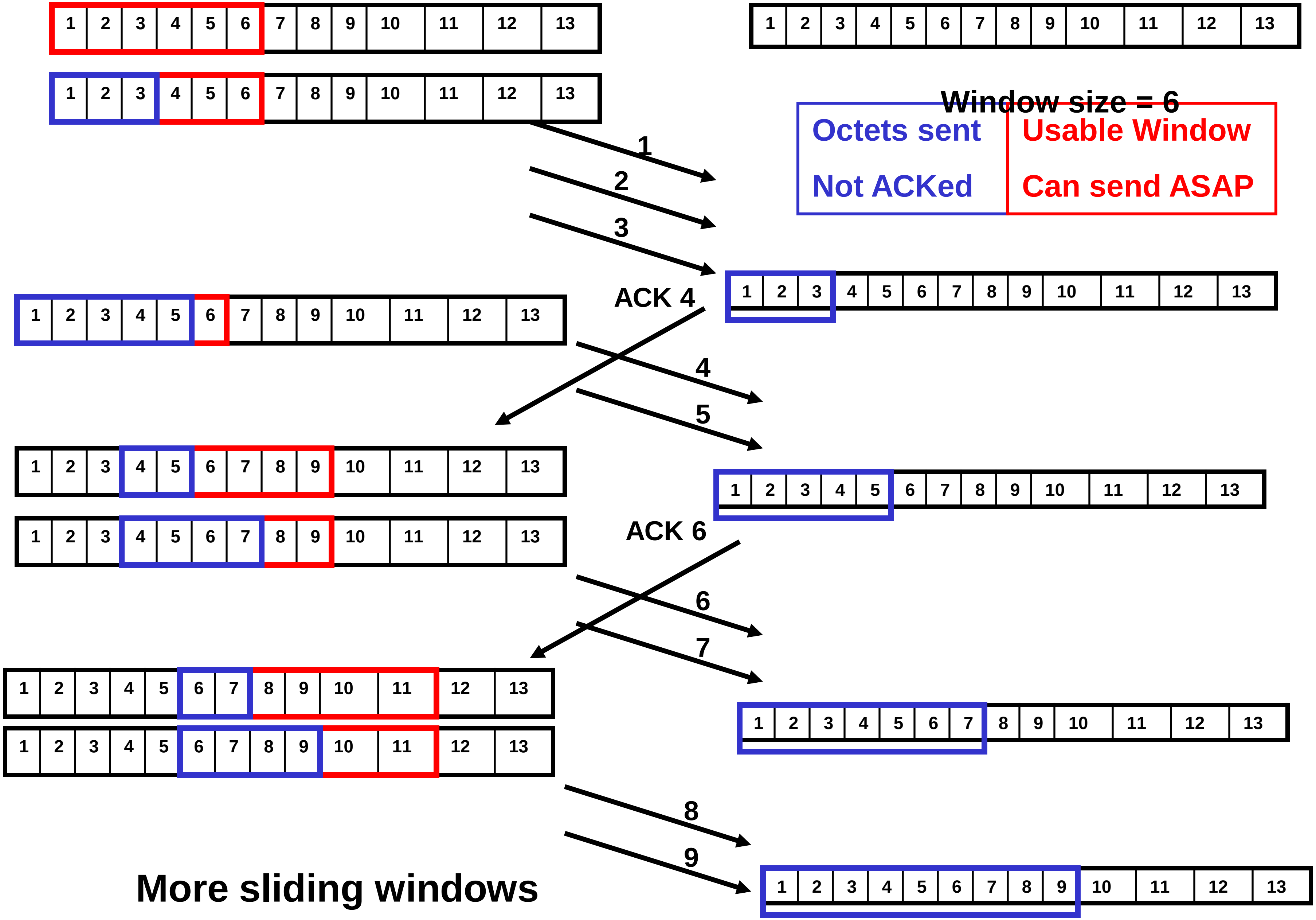
Window size = 6

Octets sent Not ACKed	Usable Window Can send ASAP

- Host A does not have to wait for an acknowledgement from Host B to keep sending data, not until the window size reaches the window size of 6, so it sends octets 4 and 5.
- Host A receives the acknowledgement of ACK 4 and can now ***slide*** its window over to equal 6 octets, 3 octets sent – not ACKed plus 3 octets which can be sent asap.

Host A - Sender

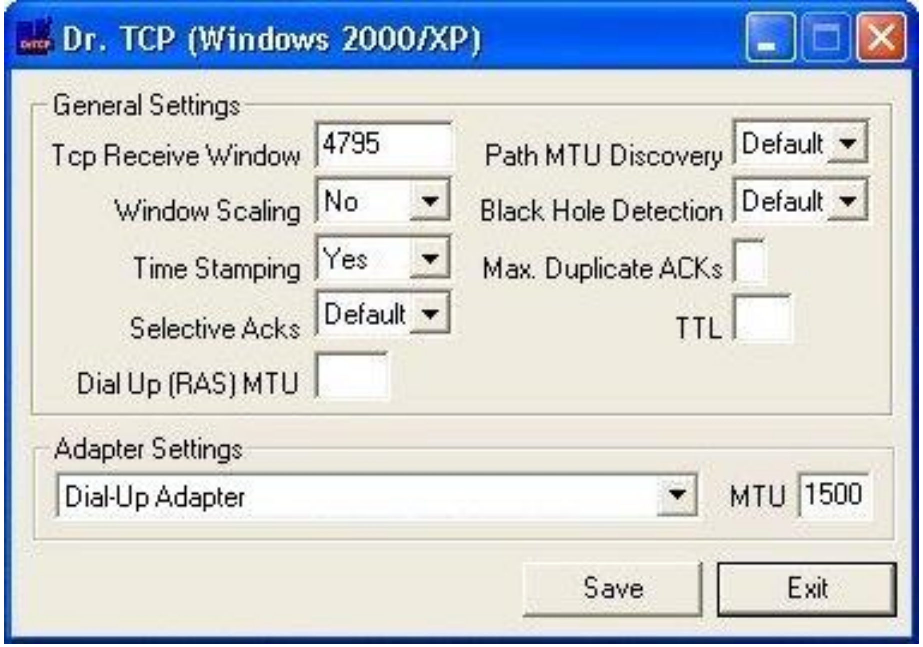
Host B - Receiver



Window size = 6

Octets sent	Usable Window
Not ACKed	Can send ASAP

More sliding windows



Dr. TCP (Windows 2000/XP)

General Settings

Tcp Receive Window: 4795

Window Scaling: No

Time Stamping: Yes

Selective Acks: Default

Dial Up (RAS) MTU:

Path MTU Discovery: Default

Black Hole Detection: Default

Max. Duplicate ACKs:

TTL:

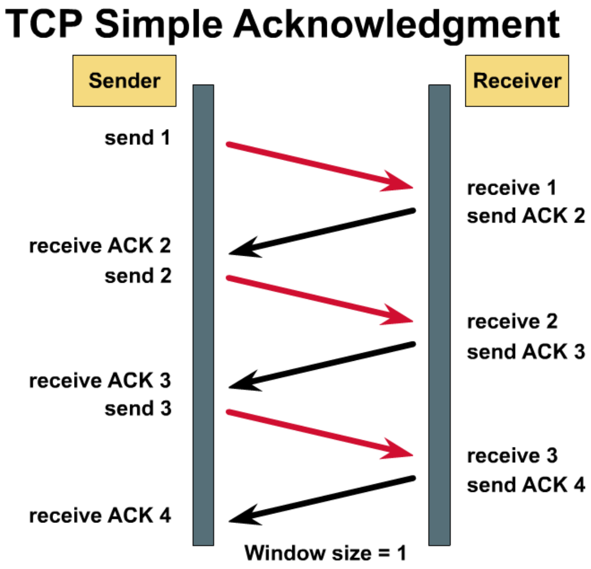
Adapter Settings

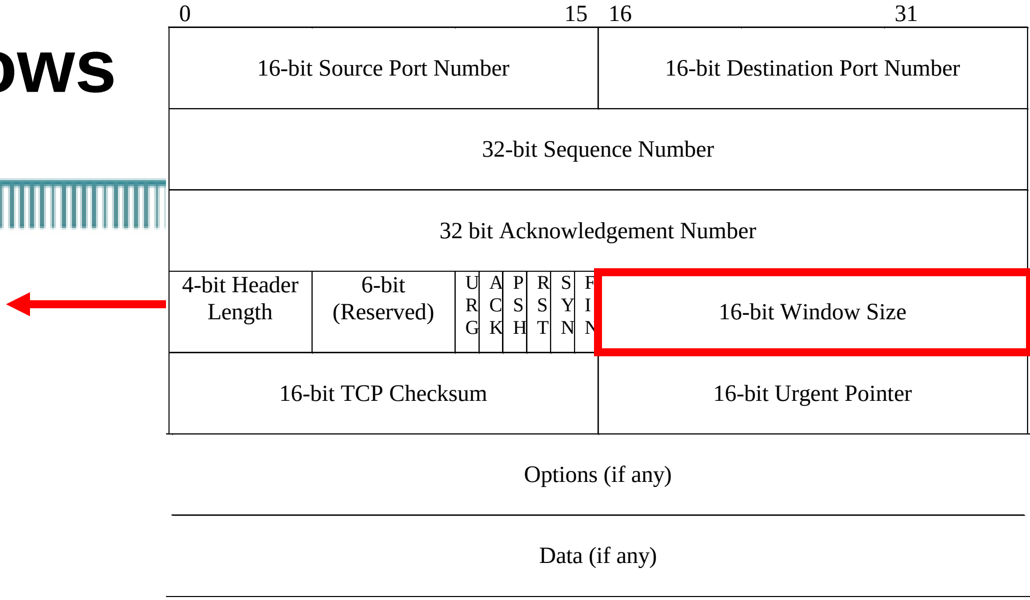
Dial-Up Adapter

MTU: 1500

- Default 8K for Windows, 32K for Linux,
- There are various unix/linux/microsoft programs that allow you to modify the default window size.
- I do not recommend that you mess around with this unless you know what you are doing.
- **"Disclaimer:** Modifying the registry can cause serious problems that may require you to reinstall your operating system. We cannot guarantee that problems resulting from modifications to the registry can be solved. Use the information provided at your own risk."
- **NOTE**: I take no responsibility for this software or any others!

Receive and Send Windows





Receive Window

- The TCP Receive Window size is the amount of receive data (in bytes) that can be buffered by this host, at one time on a connection.
- The other (sending) host can send only that amount of data before getting an acknowledgment and window update from this (the receiving) host.

Send Window

- The TCP Receive Window size of the other host.
- How much data (in bytes) that can be sent by this host before receiving an acknowledgement from the other host.

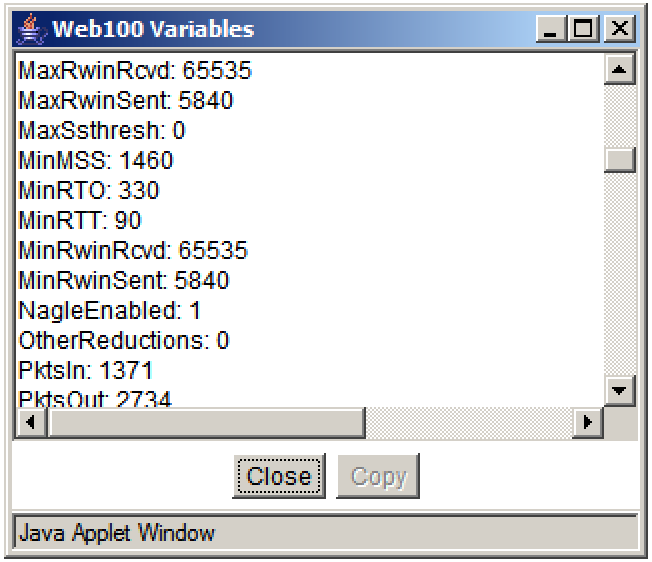
Sliding Windows – From TCPGuide.com

- Example: Server's window size was 360.
- This means the server is willing to take no more than 360 bytes at a time from the client.
- When the server receives data from the client, it places it into this buffer.
- The server must then do two distinct things with this data:
    1. **Acknowledgment:** The server must send an acknowledgment back to the client to indicate that the data was received.
    2. **Transfer:** The server must process the data, transferring it to the destination application process.
- The key point is that in the basic sliding windows system, data is acknowledged when received, but ***not necessarily*** immediately transferred out of the buffer.
- This means that is possible for the buffer to fill up with received data faster than the receiving TCP can empty it.
- When this occurs, the receiving device may need to adjust window size to prevent the buffer from being overloaded.

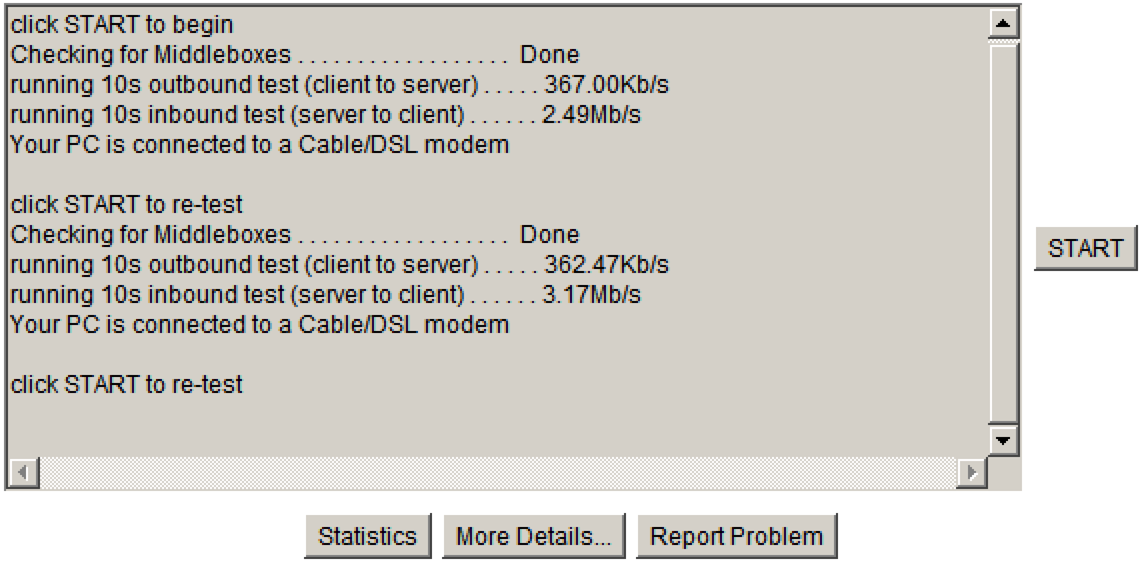
Bandwidth Testing and Other Statistics

Using a browser go to this link and click on **start**:

- http://miranda.ctd.anl.gov:7123/

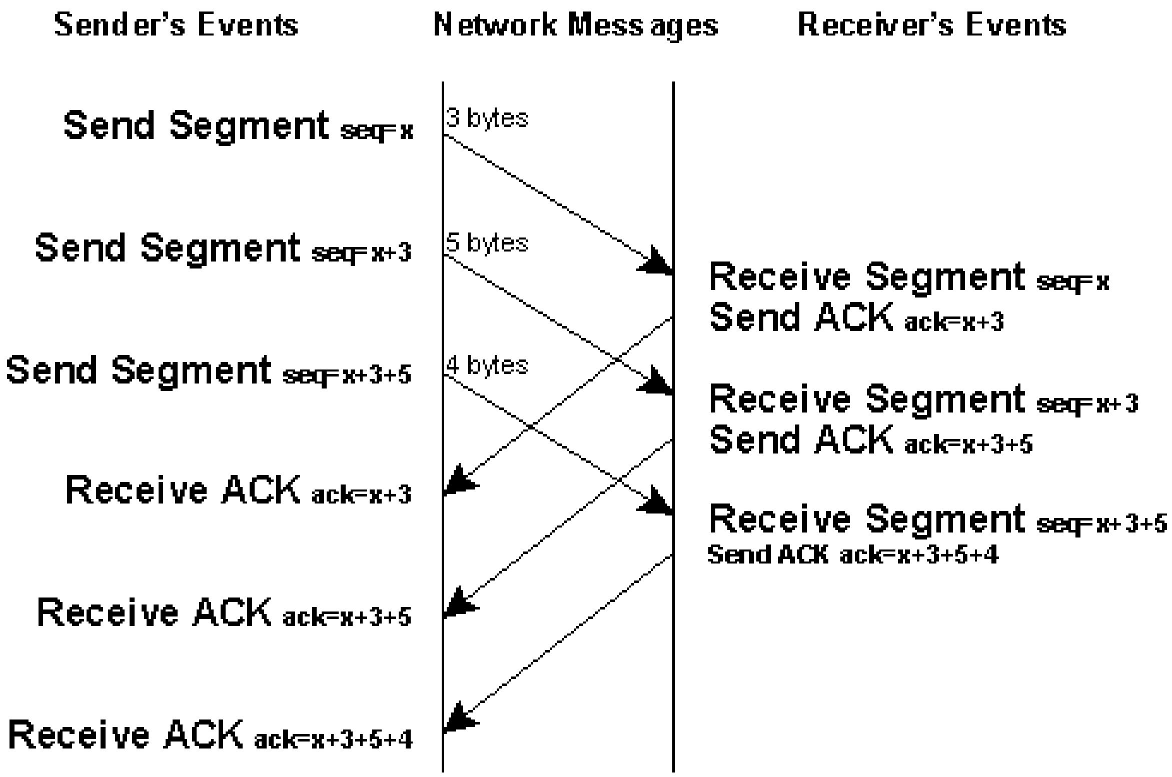


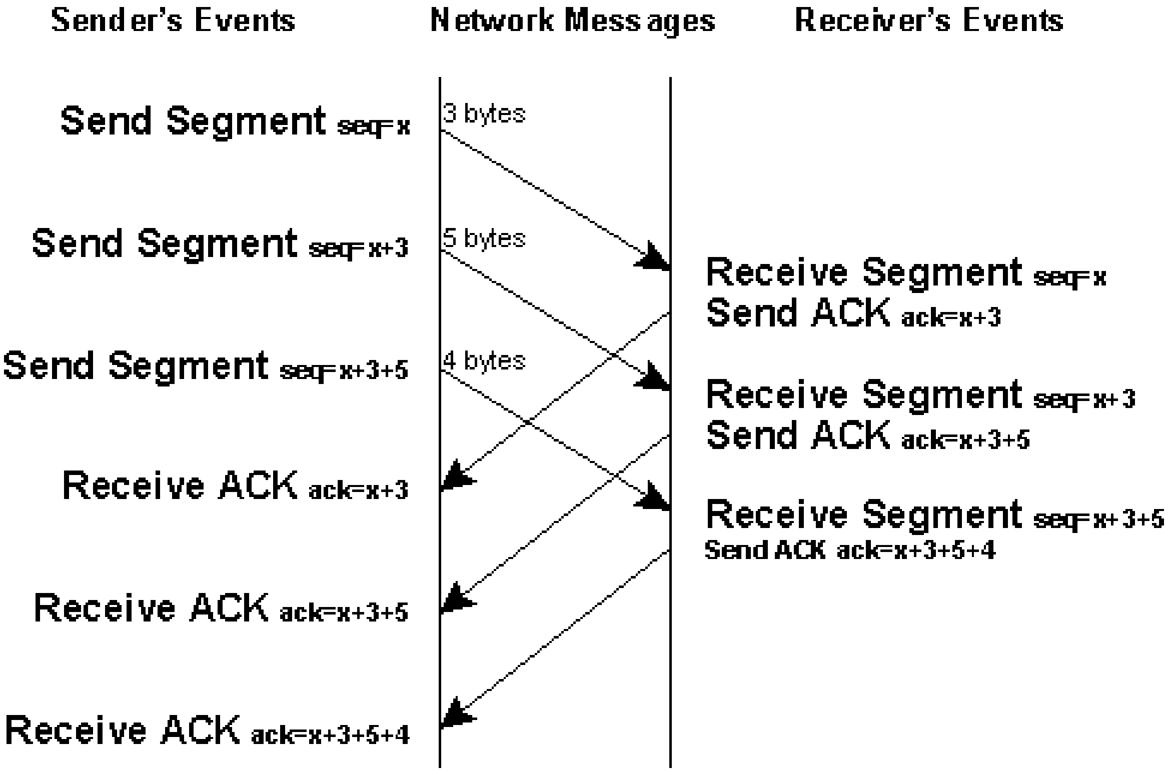
```
MaxRwinRcvd: 65535
MaxRwinSent: 5840
MaxSsthresh: 0
MinMSS: 1460
MinRTO: 330
MinRTT: 90
MinRwinRcvd: 65535
MinRwinSent: 5840
NagleEnabled: 1
OtherReductions: 0
PktsIn: 1371
PktsOut: 2734
```



```
click START to begin
Checking for Middleboxes . . . . . . . . . . . . . . . . . .  Done
running 10s outbound test (client to server) . . . . . 367.00Kb/s
running 10s inbound test (server to client) . . . . . . 2.49Mb/s
Your PC is connected to a Cable/DSL modem

click START to re-test
Checking for Middleboxes . . . . . . . . . . . . . . . . . .  Done
running 10s outbound test (client to server) . . . . . 362.47Kb/s
running 10s inbound test (server to client) . . . . . . 3.17Mb/s
Your PC is connected to a Cable/DSL modem

click START to re-test
```

More on TCP Sequence Numbers and Acknowledgements

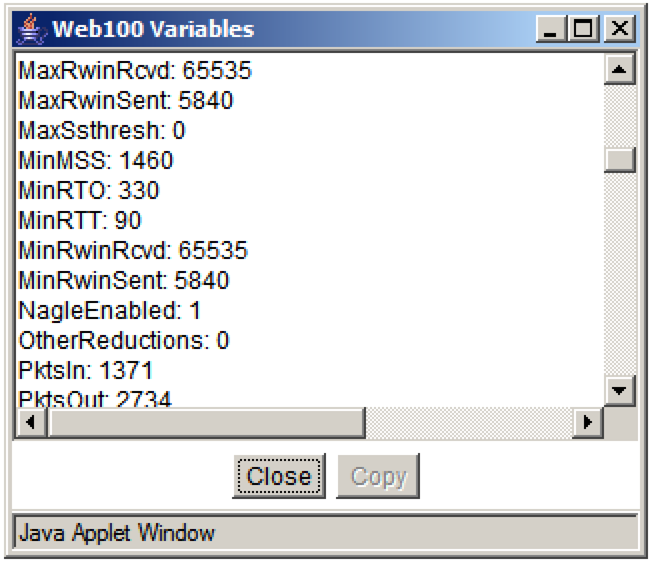




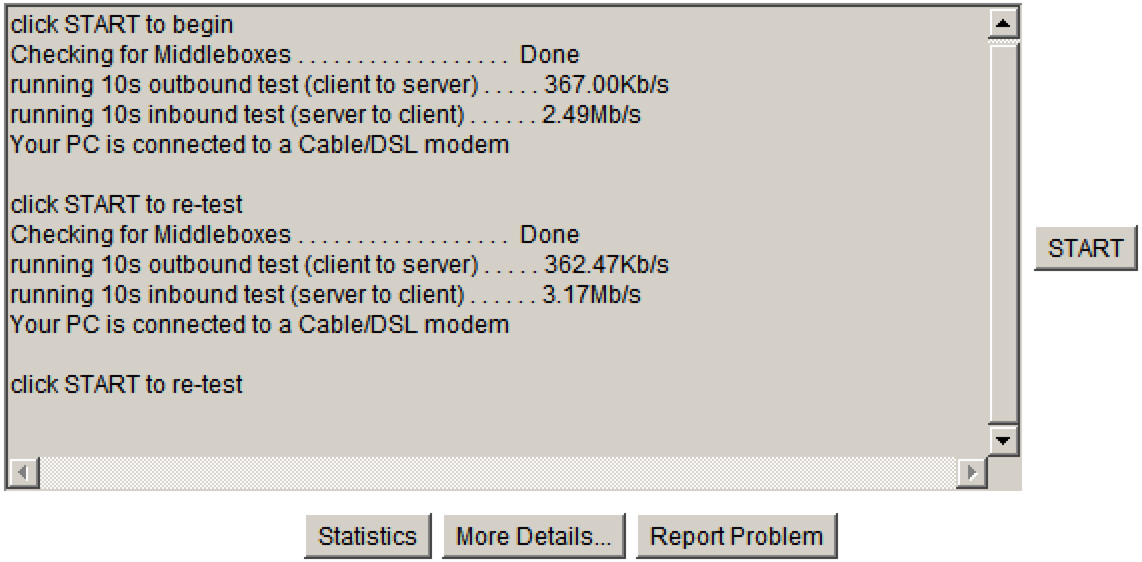
- The sequence and acknowledgment numbers are directional, which means that the communication occurs in both directions.
- The figure illustrates the communication going in one direction.
- The sequence and acknowledgments take place with the sender on the right.
- TCP provides **full-duplex service**, which means data can be flowing in each direction, independent of the other direction.
- Window sizes, sequence numbers and acknowledgment numbers are independent of each other's data flow.

We will do this in the lab:

- http://miranda.ctd.anl.gov:7123/

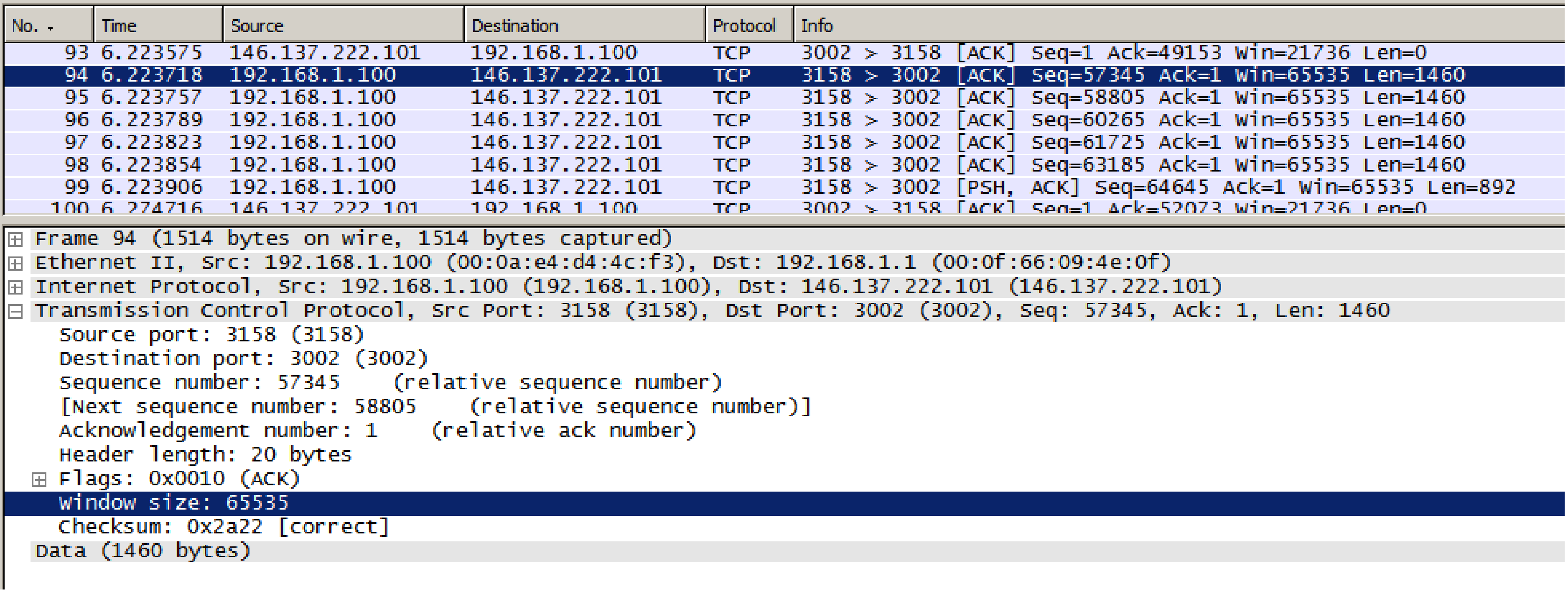


```
MaxRwinRcvd: 65535
MaxRwinSent: 5840
MaxSsthresh: 0
MinMSS: 1460
MinRTO: 330
MinRTT: 90
MinRwinRcvd: 65535
MinRwinSent: 5840
NagleEnabled: 1
OtherReductions: 0
PktsIn: 1371
PktsOut: 2734
```



```
click START to begin
Checking for Middleboxes . . . . . . . . . . . . . . . . . .  Done
running 10s outbound test (client to server) . . . . . 367.00Kb/s
running 10s inbound test (server to client) . . . . . . 2.49Mb/s
Your PC is connected to a Cable/DSL modem

click START to re-test
Checking for Middleboxes . . . . . . . . . . . . . . . . . .  Done
running 10s outbound test (client to server) . . . . . 362.47Kb/s
running 10s inbound test (server to client) . . . . . . 3.17Mb/s
Your PC is connected to a Cable/DSL modem

click START to re-test
```

Viewing Receive Window Sizes in Ethereal



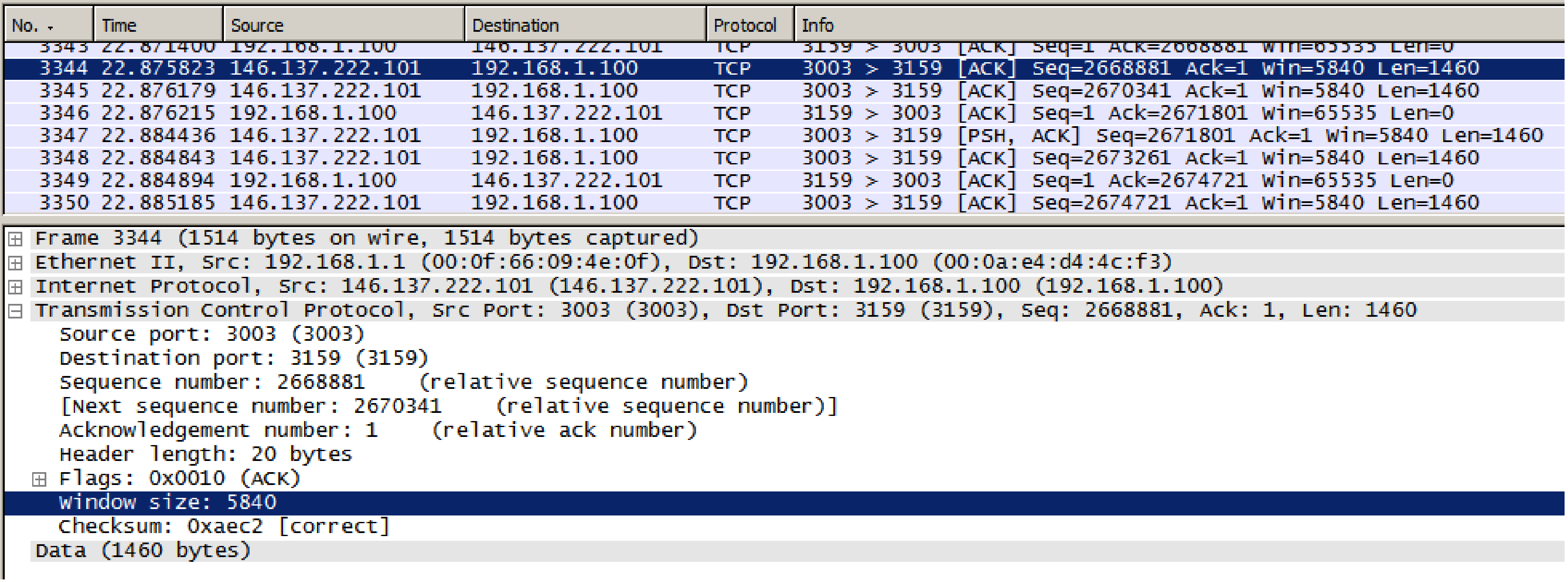
```
No. .   Time        Source            Destination       Protocol  Info
   93 6.223575  146.137.222.101   192.168.1.100     TCP       3002 > 3158 [ACK] Seq=1 Ack=49153 Win=21736 Len=0
   94 6.223718  192.168.1.100     146.137.222.101   TCP       3158 > 3002 [ACK] Seq=57345 Ack=1 Win=65535 Len=1460
   95 6.223757  192.168.1.100     146.137.222.101   TCP       3158 > 3002 [ACK] Seq=58805 Ack=1 Win=65535 Len=1460
   96 6.223789  192.168.1.100     146.137.222.101   TCP       3158 > 3002 [ACK] Seq=60265 Ack=1 Win=65535 Len=1460
   97 6.223823  192.168.1.100     146.137.222.101   TCP       3158 > 3002 [ACK] Seq=61725 Ack=1 Win=65535 Len=1460
   98 6.223854  192.168.1.100     146.137.222.101   TCP       3158 > 3002 [ACK] Seq=63185 Ack=1 Win=65535 Len=1460
   99 6.223906  192.168.1.100     146.137.222.101   TCP       3158 > 3002 [PSH, ACK] Seq=64645 Ack=1 Win=65535 Len=892
  100 6.274716  146.137.222.101   192.168.1.100     TCP       3002 > 3158 [ACK] Seq=1 Ack=52073 Win=21736 Len=0
```

```
⊞ Frame 94 (1514 bytes on wire, 1514 bytes captured)
⊞ Ethernet II, Src: 192.168.1.100 (00:0a:e4:d4:4c:f3), Dst: 192.168.1.1 (00:0f:66:09:4e:0f)
⊞ Internet Protocol, Src: 192.168.1.100 (192.168.1.100), Dst: 146.137.222.101 (146.137.222.101)
⊟ Transmission Control Protocol, Src Port: 3158 (3158), Dst Port: 3002 (3002), Seq: 57345, Ack: 1, Len: 1460
    Source port: 3158 (3158)
    Destination port: 3002 (3002)
    Sequence number: 57345    (relative sequence number)
    [Next sequence number: 58805    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x0010 (ACK)
    Window size: 65535
    Checksum: 0x2a22 [correct]
  Data (1460 bytes)
```

Receive Window

- The TCP Receive Window size is the amount of receive data (in bytes) that can be buffered by this host, at one time on a connection.
- The other (sending) host can send only that amount of data before getting an acknowledgment and window update from this (the receiving) host.

Viewing the Send Window Sizes in Ethereal



```
No. -  Time       Source           Destination      Protocol  Info
3343 22.871400 192.168.1.100   146.137.222.101 TCP      3159 > 3003 [ACK] Seq=1 Ack=2668881 Win=65535 Len=0
3344 22.875823 146.137.222.101 192.168.1.100   TCP      3003 > 3159 [ACK] Seq=2668881 Ack=1 Win=5840 Len=1460
3345 22.876179 146.137.222.101 192.168.1.100   TCP      3003 > 3159 [ACK] Seq=2670341 Ack=1 Win=5840 Len=1460
3346 22.876215 192.168.1.100   146.137.222.101 TCP      3159 > 3003 [ACK] Seq=1 Ack=2671801 Win=65535 Len=0
3347 22.884436 146.137.222.101 192.168.1.100   TCP      3003 > 3159 [PSH, ACK] Seq=2671801 Ack=1 Win=5840 Len=1460
3348 22.884843 146.137.222.101 192.168.1.100   TCP      3003 > 3159 [ACK] Seq=2673261 Ack=1 Win=5840 Len=1460
3349 22.884894 192.168.1.100   146.137.222.101 TCP      3159 > 3003 [ACK] Seq=1 Ack=2674721 Win=65535 Len=0
3350 22.885185 146.137.222.101 192.168.1.100   TCP      3003 > 3159 [ACK] Seq=2674721 Ack=1 Win=5840 Len=1460
```

```
⊞ Frame 3344 (1514 bytes on wire, 1514 bytes captured)
⊞ Ethernet II, Src: 192.168.1.1 (00:0f:66:09:4e:0f), Dst: 192.168.1.100 (00:0a:e4:d4:4c:f3)
⊞ Internet Protocol, Src: 146.137.222.101 (146.137.222.101), Dst: 192.168.1.100 (192.168.1.100)
⊟ Transmission Control Protocol, Src Port: 3003 (3003), Dst Port: 3159 (3159), Seq: 2668881, Ack: 1, Len: 1460
    Source port: 3003 (3003)
    Destination port: 3159 (3159)
    Sequence number: 2668881    (relative sequence number)
    [Next sequence number: 2670341    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x0010 (ACK)
    Window size: 5840
    Checksum: 0xaec2 [correct]
  Data (1460 bytes)
```

Send Window

- The TCP Receive Window size of the other host.
- How much data (in bytes) that can be sent by this host before receiving an acknowledgement from the other host.

Ch.11 – TCP/IP Transport and Application Layers

Cabrillo College

CIS 81 and CST 311

Rick Graziani

Spring 2006